RESOURCE AND PATIENT MANAGEMENT SYSTEM

FILEMAN 22
(DI)

Patch Notes

Version 22
April 2003

Information Technology Support Center
Division of Information Resources
Albuquerque, New Mexico

TABLE OF CONTENTS

1.0 DI*22*1 SEQ #1: Post Verification Fixes

1.1 Description

After V22 of VA FileMan was released to National VistA Support (NVS) for field distribution several bugs were discovered at the test sites that could affect various packages. The following is list of the routine(s) and the problem that was corrected:

1.1.1 Routine: DICATTD

Bug: When using the option Modify File Attributes AND used the Screen-Mode version AND were adding a Free Text multiple, the 2nd piece of the 0 node of the DD would be corrupted.

1.1.2 Routine: DICOMPV

Bug: It was discovered that when using a Backward Extended Pointer and the file that was being jumped to had a DINUMed .01 field back to the originating file, a jump was not being allowed.

1.1.3 Routine: DIC3

Bug: Input variable DIC("S") to the ^DIC routine altered local variable Y. FileMan was not protecting Y and expected it to contain the record number just looked-up, so the wrong value was displayed for the .01 field when entries were presented to the end-user.

1.1.4 Routines: DIC1, DIC11

Bug: When the internal value of a date, pointer or variable pointer is passed as a default lookup value in input variable DIC("B") to the ^DIC routine, it should be displayed to the end-user in external format. This worked in V21 but didn't in V22.

1.1.5 Routine: DIE

Bug: After a call to ^DIE with DR="[template name]", a ^TMP("DIE",number) node was not being killed.

1.1.6 Routine: DIEZ

Bug: An undefined X variable error could occur during a KIDS installation, when KIDS called FileMan to uncompile an input template.

1.1.7 Routines: DICA, DICA1

Bug: If an UPDATE^DIE call is made from within another UPDATE^DIE call (for example, when the Updater executes a cross-reference that makes another Updater call), the call could fail and Dialog file error messages generated. Also the variable D was leaking out of UPDATE^DIE.

1.1.8 Routine: DIEFW

Bug: An undefined DIEFLOCK could occur if the Filer was called with the "K" flag to do locking, and within that Filer call, a call was made to WP^DIE (from within a cross-reference, for example) without the "K" flag.

1.1.9 Routines: DIFROMSX, DIFROMXY

Change: Before, if a partial DD was sent with KIDS and the file had KEYs, all of the field definitions for the KEY fields had to be sent. We decided that wasn't necessary, so after this patch, the field definition for a key field must be sent only if the key is compound and other field(s) in the key are being sent.

1.1.10 Routines: DIK1, DIKZ1

Bug: The ^DIK entry points did not have time-outs on the Lock commands they issued.

1.1.11 Routine: DIR1

Bug: MAS patch DG*5.3*149 brought new code into routine DGREG. The old code directly executed the input transform from subfile 2.101, field .01 to check the validity of a value entered by the user (in X). The new code instead calls CHK^DIE (which also executes the input transform) to check the validity of the value. The input transform code for this field sets the variable DINUM. The MAS code at REG+2^DGREG then uses DINUM. In version 22, FileMan NEWs DINUM in the CHK^DIE call before executing the input transform, because DINUM is not a documented output variable, and because it is a dangerous variable to leave around. MAS does not have time to patch their code before the planned release of version 22, so FileMan will temporarily 'leak' DINUM in order to avoid a hard error. After MAS patches their code, FileMan will issue another patch to restore the NEW DINUM.

1.1.12 Routines: DIE1, DIEF, DIKC

Bug: If there is an X1 array element corresponding to the .01 field in the new-style Index logic or condition, make sure it is set to null if the record is being added.

1.1.13 Routines: DINIT2A1, DINIT2A2, DINIT2A3, DINIT2A4

Bug: These DINIT routines fix a typo on the description of the following field in the Cross-Reference Values multiple of the Index file (#.11):

11.1 Cross-Reference Values (Multiple-.114)
4.5 Computed Code
was: "Answer with M code that set X..."
should be: "Answer with M code that sets X..."

The DINIT routines also add information about the X1 and X2 arrays to the descriptions of the following fields in the Index file (#.11):

1.1 Set Logic
1.4 Set Condition Code
2.1 Kill Logic
2.4 Kill Condition Code

For each of the above fields, the following paragraph was appended to the description:

When fields that make up a cross-reference are edited and the kill and set logic are executed, the X1(order#) array contains the old field values, and the X2(order#) array contains the new field values. If a record is being added, and there is an X1(order#) array element that corresponds to the .01 field, it is set to null. When a record is deleted, all X2(order#) array elements are null.

Since the KIDS build for this patch brings in these fields automatically, there is no need to rerun DINIT after installing this patch.

2.0 DI*22*13 SEQ #2: DIC Lookup On Variable Pointer By IEN

2.1 Description

2.1.1 Routine: DIC11

In version 21 of FileMan, if lookup value is accent grave followed by a number, FileMan first looks on the current file for a matching IEN. If none is found, but we can add records to the file, and if the .01 field is a pointer or variable pointer, FileMan looks on the pointed-to file for a matching IEN. This did not work for variable pointers in version 22 because it was felt that the user had no control over which of the pointed-to files might be used for the match. However, on the NOIS, the developers had used DIC("V") to limit the lookup to only one of the pointed-to files. This patch makes the lookup work on variable pointers as well as pointers.

3.0 DI*22*12 SEQ #3: New API To Delete A Cross-Reference

3.1 Description

This patch introduces two new APIs in VA FileMan, DELIX^DDMOD and DELIXN^DDMOD. DELIX^DDMOD can be used to delete a Traditional Cross-Reference definition from the data dictionary. DELIXN^DDMOD can be used to delete a New-Style Index definition from the Index file (#.11). Optionally, depending on the type of cross-reference deleted, the APIs will also delete the data in the index or execute the kill logic for all entries in the file. Compiled input templates that contain the field on which the cross-reference is defined are recompiled. If cross-references on the file are compiled, they are also recompiled.

3.2 Documentation

3.2.1 DELIX^DDMOD: Traditional Cross-Reference Deleter

This procedure deletes a Traditional Cross-Reference definition from the data dictionary. Optionally, it deletes the data in the index or executes the kill logic for all entries in the file. Compiled input templates that contain the field on which the cross-reference is defined are recompiled. If cross-references on the file are compiled, they are recompiled.

3.2.1.1 Format

DELIX^DDMOD(FILE,FIELD,CROSS_REF,FLAGS,OUTPUT_ROOT,MSG_ROOT)

3.2.1.2 Input Parameters

FILE (Required) File or subfile number.

FIELD (Required) Field number.

CROSS_REF (Required) Cross-reference number. Traditional cross-references are defined in the data dictionary under ^DD(file#,field#,1,cross reference number).

FLAGS (Optional) Flags to control processing. The possible values are:

K For Regular, KWIC, Mnemonic, and Soundex-type cross-references, delete the data in the index.

For MUMPS and Trigger-type cross-references, execute the kill logic of the cross-reference for all entries in the file. For Bulletin-type cross-

references, the "K" flag is ignored; the kill logic for Bulletin-type cross-references is never executed by this procedure.

W Write messages to the current device as the index is deleted and cross-references and input templates are recompiled.

OUTPUT_ROOT (Optional) The name of the array that should receive information about input templates and cross-references that may have been recompiled and a flag to indicate that the deletion was audited in the DD Audit file (#.6). See Output below. This must be a closed root, either local or global.

MSG_ROOT (Optional) The name of the array that should receive any error messages. This must be a closed root, either local or global. If not passed, errors are returned descendent from ^TMP("DIERR",$J).

3.2.1.3 Output

OUTPUT_ROOT See OUTPUT_ROOT under Input Parameters.

If the field on which the deleted cross-reference was defined is used in any compiled input templates, those input templates are recompiled. Information about the recompiled input templates is stored descendant from OUTPUT_ROOT("DIEZ"):

OUTPUT_ROOT("DIEZ",input template #) = input template name ^ file # ^ compiled routine name

If cross-references for the file are compiled, they are recompiled, and the compiled routine name is stored in OUTPUT_ROOT("DIKZ"):

OUTPUT_ROOT("DIKZ") = compiled routine name

If the data dictionary for the file is audited, an entry is made in the DD Audit file (#.6) and OUTPUT_ROOT("DDAUD") is set to 1:

OUTPUT_ROOT("DDAUD") = 1

3.2.1.4 Example

1. In this example, regular cross-reference #4 (the "C" index), defined on field #12 in file #16200, is deleted. The "K" flag indicates that the entire ^DIZ(16200,"C") index is removed from the file.

```
>D DELIX^DDMOD(16200,12,4,"K","MYOUT")

>ZW MYOUT

      MYOUT("DDAUD")=1
      MYOUT("DIEZ",100)=ZZTEST EDIT^16200^ZZIT
      MYOUT("DIKZ")=ZZCR
```

The MYOUT output array indicates that the deletion was recorded in the DD Audit file (#.6). Field #12 is included in the compiled input template ZZTEST EDIT (#100), which is compiled into the ZZIT namespaced routines. Cross-references on file #16200 are compiled under the ZZCR namespace.

2. In this example, the whole-file regular cross-reference #7 (the "N" index), defined on field #15 within subfile #16200.075, is deleted.

The "K" flag indicates that the entire ^DIZ(16200,"N") index should be removed, and the "W" flag indicates that messages should be printed to the current device.

```
      >D DELIX(16200.075,15,7,"KW")

      Removing index ...
      Deleting cross-reference definition ...


      Compiling ZZ TEST CR Input Template of File 16200...
      'ZZIT1' ROUTINE FILED..
      'ZZIT' ROUTINE FILED....
      'ZZIT2' ROUTINE FILED.

      Compiling Cross-Reference(s) 16200 of File 16200.

      ...SORRY, HOLD ON...

      'ZZCR1' ROUTINE FILED.
      'ZZCR2' ROUTINE FILED.
      'ZZCR3' ROUTINE FILED.
      'ZZCR4' ROUTINE FILED.
      'ZZCR5' ROUTINE FILED.
      'ZZCR' ROUTINE FILED.
```

3.2.1.5 Error Codes Returned

202 The specified parameter is missing or invalid.

301 The passed flags are incorrect.

401 The file does not exist.

406 The file has no .01 definition.

407 A word-processing field is not a file.

501 The file does not contain the specified field.

3.2.2 DELIXN^DDMOD: New-Style Index Deleter

This procedure deletes a New-Style Index definition from the Index file. Optionally, it deletes the data in the index or executes the kill logic for all entries in the file. Compiled input templates that contain one or more of the fields defined in the index are recompiled. If cross-references on the file are compiled, they are recompiled.

3.2.2.1 Format

```
DELIXN^DDMOD(FILE,INDEX,FLAGS,OUTPUT_ROOT,MSG_ROOT)
```

3.2.2.2 Input Parameters

FILE (Required) File or subfile number. For whole-file indexes, this is the number of the file at the upper level where the data in the index resides.

INDEX (Required) Index name.

FLAGS (Optional) Flags to control processing. The possible values are:

K For Regular indexes, delete the data in the index.

For MUMPS indexes, execute the kill logic for all entries in the file.

W Write messages to the current device as the index is deleted and cross-references and input templates are recompiled.

OUTPUT_ROOT (Optional) The name of the array that should receive information about input templates and cross-references that may have been recompiled. See Output below. This must be a closed root, either local or global.

MSG_ROOT (Optional) The name of the array that should receive any error messages. This must be a closed root, either local or global. If not passed, errors are returned descendent from ^TMP("DIERR",$J).

3.2.2.3 Output

OUTPUT_ROOT See OUTPUT_ROOT under Input Parameters. If a field used in the index is used in any compiled input templates, those input templates are recompiled. Information about the recompiled input templates is stored descendant from OUTPUT_ROOT("DIEZ"):

OUTPUT_ROOT("DIEZ",input template #) = input template name ^ file # ^ compiled routine name

If cross-references for the file are compiled, they are recompiled, and the compiled routine name is stored in OUTPUT_ROOT("DIKZ"):

OUTPUT_ROOT("DIKZ") = compiled routine name

3.2.2.4 Example

1. In this example, the new-style "G" index defined on file #16200 is deleted. The "K" flag indicates that the entire ^DIZ(16200,"G") index should be removed from the file.

```
>D DELIXN^DDMOD(16200,"G","K","MYOUT")

>ZW MYOUT
MYOUT("DIEZ",94)=ZZ TEST^16200^ZZIT
MYOUT("DIEZ",100)=ZZ TEST A^16200^ZZITA
MYOUT("DIKZ")=ZZCR
```

The MYOUT output array indicates that a field or fields defined in the deleted index are used in the compiled input templates ZZ TEST (#94) and ZZ TEST 2 (#100). Those two input templates were recompiled. Cross-references on file #16200 were also recompiled under the ZZCR namespace.

2. In this example, the whole-file regular index (the "J" index) is deleted. The fields in the index come from fields in a multiple, subfile #16200.075, but the whole-file index resides at the top-level file #16200. The "K" flag indicates that the entire ^DIZ(16200,"J") index should be removed, and the "W" flag indicates that messages should be printed to the current device.

```
D DELIXN^DDMOD(16200,"J","KW","MYOUT")

Removing index ...
Deleting index definition ...


Compiling ZZ TEST Input Template of File 16200....
'ZZIT' ROUTINE FILED....
'ZZIT1' ROUTINE FILED.


Compiling ZZ TEST A Input Template of File 16200....
'ZZITA' ROUTINE FILED....
'ZZITA' ROUTINE FILED.

Compiling Cross-Reference(s) 16200 of File 16200.

...SORRY, JUST A MOMENT PLEASE...

'ZZCR1' ROUTINE FILED.
'ZZCR2' ROUTINE FILED.
'ZZCR3' ROUTINE FILED.
'ZZCR4' ROUTINE FILED.
'ZZCR5' ROUTINE FILED.
'ZZCR6' ROUTINE FILED.
'ZZCR7' ROUTINE FILED.
'ZZCR8' ROUTINE FILED.
'ZZCR9' ROUTINE FILED.
'ZZCR10' ROUTINE FILED.
'ZZCR' ROUTINE FILED.
```

3.2.2.5 Error Codes Returned

202 The specified parameter is missing or invalid.

301 The passed flags are incorrect.

4.0 DI*22*14 SEQ #4: %DT Enhancements/Fixes For Y2K

4.1 Description

This patch fixes some bugs, and contains an enhancement to address a Y2K issue.

4.1.1 Routines: DIDT (%DT), DIDTC (%DTC)

4.1.2 Bug fixes

1. Input such as such as "07/00", "JUL 00", and "00 JUL", where the two-digit year is "00", was rejected even if inexact dates are allowed. With this patch, if %%DT'["X" the above inputs are interpreted as July 2000.

2. Input such as "0700", where the input is exactly 4 numerics with "00" as the last to digits, was interpreted as July of the current year (if the "P" and "F" flags weren't used). With this patch, if %DT'["X", "0700" is interpreted as July 2000; if %DT["X", it is rejected.

3. Input such as "070", where the input is exactly 3 numerics, and the first two numbers are a valid month and the third number is a 0, was interpreted as July of the current year (if the "P" and "F" flags weren't used). With this patch, "070" is rejected.

4.1.3 Enhancement

Introduce a new "M" flag for %DT. This flag allows input such as "0701", "07/01", "JUL 01", "01 JUL" to be interpreted as July 2001, instead of July 1st.

M Only Month and year input is allowed (example 1). If only a month and two digits are entered, interpret the two digits as a year instead of a day (example 2).

If the M flag is used with the X flag, a month must be specified; otherwise, the input can be just a year (example 3).

M Flag

Ex.	Input	Date Returned	Date Returned Without M
1)	7-05-2005	--invalid--	July 5, 2005
2)	7-05	July 2005	July 5, 2000*

* Assumes the current year is 2000 and the P and F flags aren't used.

```
M Flag with X
================================================
                    Date Returned   Date Returned
Ex.   Input         With X          Without X
------------------------------------------------
3)    05 or 2005    --invalid--     2005
================================================
```

5.0 DI*22*7 SEQ #5: Data Dictionary Reports/ Definition

5.1 Description

5.1.1 Routines: DID1, DID2

Bug: When running the List File Attributes option on a file that has extensive M code for Cross References, Input Transforms and the like, if the Right Margin was small enough, it was possible to go into an endless loop. This was noticed when trying to List File Attributes for file #63 and sending the output to the Browser with a right margin less than 132.

The result was a "disk full" error.

5.1.2 Routine: DID1

Bug: In a Standard DD Listing, the Descriptions of cross-references weren't formatted correctly within the margins of the report.

5.1.3 Routines: DIV, DIVR

Bug: If the UTILITY FUNCTIONS/VERIFY FIELDS option is used, the generated report could scroll off the screen. This patch makes the option prompt for device, and if the report is printed to the screen, makes it issue the "Enter RETURN to continue or '^' to exit:" prompt at the end of each page.

5.1.4 Routines: DIVR, DIVR1

Bug: When the UTILITY FUNCTIONS/VERIFY FIELDS option is used, and if a required or key field equals only one or more spaces, the message "Missing" would be printed. This patch changes the message to "Equals only 1 or more spaces" to more accurately reflect the reason an entry is flagged as having a possible problem.

5.1.5 Routine: DICATT

Bug: The prompt after option 4, Modify File Attributes, had a typo: the word "USER" should be "USE":

```
DO YOU WANT TO USER THE SCREEN-MODE VERSION? Yes//
```

5.1.6 Routines: DIPR7, DINIT11B

Bug: Change field 20 DEVELOPER in file 1 (FILE) so that LAYGO is not allowed to file 200 (NEW PERSON file). This should have never been allowed and was an

oversite. DIPR7 is a pre-init routine that changes the second piece of the field from 200P to 200P'. DINIT11B is the routine that is the part of the initialization of FileMan that brings in field 20.

****NOTE**** You may safely delete routine DIPR7 after installing this patch.

6.0 DI*22*16 SEQ #6: Allocation Errors In Lookup (DIC) Call

6.1 Description

This patch repairs allocation errors when doing a call to the FileMan lookup routine ^DIC.

6.1.1 Routine: DIC3

Repairs allocation error at C4+2^DICUIX2. This happened because one of the Mailman files had some invalid nodes that looked like cross-references. Each node had a subscript that contained an "R" followed by a number. This error would only happen on a real file if it had several thousand individual cross-references (which would not be likely to happen). Nois: KAN-0999-40493.

6.1.2 Routines: DICM0,DICM3

Repairs allocation error at S+14^DIC3. This would only happen when a cross-reference was on a pointer or variable pointer field, there were many (over 1000) entries that pointed to an entry whose .01 field matched the value being looked-up, but a screen made it so that most of the entries with the pointer were not selected. Noises: HIN-0999-42903, ATG-0999-32715, SLC-0999-52640, DAY-0899-41847.

7.0 DI*22*15 SEQ #7: Comma in 1st Position NULLSUBSCR

7.1 Description

7.1.1 Routine: DICF1

A problem was reported when using $$FIND1^DIC and the first position of the [.]VALUE was equal to a Comma, a Null Subscript error would occurr at LOOP+3^DICFIX. For example, VALUE=",DOE,JANE"

8.0 DI*22*10 SEQ #8: Slow Record Deletion

8.1 Description

8.1.1 Routine: ^DIK1

Bug: Suppose ^DIK is used to delete a record with an internal entry number (ien) of X in a file, X equals what's in the 3rd piece of the header node of the file, and there is no record in the file with an ien of X-1. If either of the following is true:

- X is the last ien used in the file, and the difference between X and the ien of the preceding record is large; or

- the difference between X and the ien of the next record is large

^DIK could go into an extremely time-consuming loop as it attempts to find the ien of the last (or next) record in the file.

For example, file 16010 has the following header node and entries:

```
^DIZ(16010,0) = TEST^16010^xxx^3
                          ^------- 3rd piece of header node
^DIZ(16010,1,0) = FIRST
^DIZ(16010,10,0) = SECOND
^DIZ(16010,12345678901234,0) = THIRD
```

If you use ^DIK to delete entry #10 (and xxx = 10) or entry #12345678901234 (and xxx = 12345678901234), ^DIK could go into a time-consuming loop.

9.0 DI*22*4 SEQ #9: Misc. DIC Fixes

9.1 Patch Description

This patch fixed a few minor bugs and adds some enhancements to correct deficiencies in the FileMan Lookup routines ^DIC*, Finder routines ^DICF*, and Editing routines ^DIE*.

Y2K Waiver Exemption ID#: Y2KWE0015

9.1.1 Routines: DIC, DIC1, DICM, DICM0, DICM3

"^^" OUT. Users will be allowed to enter a double up-arrow ^^ at a prompt in the ^DIC lookup. This will get them clear out of the lookup. Previously, users were able to enter a single up-arrow ^ but this only took them out of the search through the current index. If there were many indexes being searched, the search could continue for a long time with no way to exit.

9.1.2 Routines: DDSU, DICQ, DICQ1, DICLIX

Control number of entries displayed in "?" help. Developers needed a way to control question mark help in the ^DIC lookup so that the output didn't scroll off the screen. New input variable DIC("?N",file#)=number_of_entries_to_display can be set to the number of entries to be displayed at any one time in question mark help.

9.1.3 Routines: DIC4, DICF4, DICM0, DICM3, DICN, DICQ1, DIE,DIE1, DIE17,DIE3, DIED, DIEZ1

Control lookup to pointed-to files. During lookups on files with either indexed pointer or variable pointer fields, FileMan does a lookup on the pointed-to file, lookup for a match to the lookup value entered by the user. FileMan always looked on either the "B" index only on the pointed-to file (if DIC(0) did not contain "M"), or else on all lookup indexes. A new input array to the interactive lookup ^DIC or to the silent data base server lookup routine FIND^DIC, allows the developer to specify what indexes are to be used during lookup on a pointed-to file. The array for ^DIC looks like

DIC("PTRIX",from_file#,pointer_field#,to_file#)=list_of_indexes.

For the Finder, it looks like:

DINDEX(("PTRIX",from_file#,pointer_field#,to_file#)=list_of_indexes

For ^DIE, it looks like:

DIE("PTRIX",from_file#,pointer_field#,to_file#)=list_of_indexes

9.1.4 Routines: DIC, DIC0, DIC1, DIC2, DIC3, DIC5, DICFIX, DICM, DICM0, DICM3

Continue looking through all indexes. ^DIC lookup does an initial pass through the list of indexes, looking for matches to the lookup value as it was entered by the user. If any are found, they are presented to the user, and whether or not any are accepted, FileMan gets out. Only if it finds no matches, does it attempt transforms on the lookup, such as converting pointers, variable pointers, sets or dates to their internal format, or converting the lookup value to upper case. This can cause the lookup to miss some of the matching values. A new flag "T" can be put into DIC(0) that will cause FileMan to keep looking until it has either tried everything, or until the user selects an entry. In addition, the transforms for pointers, variable pointers, dates or sets are done during the first pass, so indexes are truly searched in the order requested by the user.

9.1.5 Routines: DICF2, DICF3, DICF4, DICF5

"O" flag in Finder call. It was discovered here that the "O" flag in the finder didn't work the same in version 22 as it did in version 21. With this patch, the behavior is the same as version 21, in that an entire pass through all indexes is made to find any exact matches, before doing a second pass to find partial matches.

9.1.6 Routine: DICN1

'Before' values in SET/KILL logic. It was discovered here that when a new entry was added using the classic interactive lookup ^DIC, the X1 and X2 array entries for the "BEFORE" values for use in the SET and KILL logic of new style indexes were not always reliable. This corrects that bug.

9.1.7 Routine: DICM

'Missing Right Parenthesis' error in Print. When doing a print, at the PRINT FIELD prompt, if a field name greater than 30 characters long was entered, along with a header qualifier in mixed case, a 'missing right parenthesis' error occurred. This error was actually happening during the lookup for field name on the DD.

9.1.8 Routine: DICM2

Variable pointer prefix in lower case. When doing a lookup on an indexed variable pointer field, if the user entered the prefix in lower case or mixed case, but the prefix was in upper case on the file, the lookup failed.

9.1.9 Routine: DICUIX2

UNDEF error in lookup. It was discovered here that an UNDEF error could occur if the code used to build index information for the classic lookup ^DIC or the Finder FIND^DIC could not determine what field was indexed.

9.1.10 Routines: DICQ, DICQ1, DICUIX1

Question mark help was very slow when a fairly large file pointed to a very large file like the PATIENT file. We were building a temporary index. The algorithm that decides when to build an index was changed. Also, in question mark help, we won't ever build a temporary index, but will instead $O through the index on the pointer, so records will not be in alphabetic sequence. This sequence was always used on pointers in V21.

9.1.11 Routine: DIC5

When a numeric lookup value is entered and indexed field is a pointer, make lookup by IEN work the way it did in version 21.

9.1.12 Routine: DIC3

Lookups on the DRUG file by synonym were selecting the right drug, but displaying the wrong .01 field value. This happened when DIC(0) contained an "O", there was more than one exact match on the SYNONYM index to the lookup value, but all but one match were screened out by a DIC("S") screen.

9.1.13 Routine: DIC

DIC("W") was being set earlier in a call to ^DIC in Version 22 than it was in Version 21. This caused an UNDEF error to occur in some files with a user-defined lookup. This change restores backwards compatibility in this area to V21.

9.1.14 Routine: DICF4

When a call to FIND^DIC was made with the "M" flag, and an entry was found on an index other than the first one in the list, an extra node was returned containing the external index value, even if the FIELDS parameter did not contain "IXE". This caused the Finder Delphi Component to display the wrong values.

9.1.15 Routine: DICA

If an UPDATE^DIE call is made and the FDA uses a LAYGO/Finding node to lookup or add an entry at the top level of a file, and an Adding node is used to add an entry into a multiple within that record, and the top level record is added because it was not found, a null subscript error would occur. For example, if the FDA is:

```
FDA(1000,"?+1,",.01)="TEST ENTRY"
FDA(1000.02,"+2,?+1,",.01)="TEST SUBENTRY"
```

and the Updater has to add TEST ENTRY to file #1000, a null subscript error will occur.

9.1.16 Routines: DIE1,DIE2

If ^DIE was used to edit a subfile directly, and that subfile was at least two levels down from the top, ^DIE was not updating the DA array correctly when the user went down into and back out of multiples within that subfile.

9.1.17 Routine: DIEQ

Set the DA array to the subfile level if the user enters ? or ?? at the Select prompt of a multiple. This ensures that xecutable help, screens on pointer or sets, and input transforms on the .01 field of the multiple can rely on the DA array to be at the subfile level, whether ? or ?? is entered at the Select prompt or at the .01 field of the multiple.

10.0 DI*22*2 SEQ #10: Sort/Print Fixes

10.1 Patch Description

10.1.1 Bug Fix

1. If a user specified a print field whose data type definition is a Date/Time and they used a date Function along with the Count(!) operator, the FileMan internal representation would be printed. For example, !MONTH(DATE FIELD)

2. Previous to this patch FileMan Sorts were limited to 7 levels. With this patch the maximum number of Sort levels has been increased to 15.

3. When computing global length and comparing it to what was set in the MUMPS OPERATING SYSTEM file, Global Subscript truncation would begin before it was necessary.

4. When a Sort contain a Boolean expression, for example, MY FIELD["A", the Boolean operator was getting concatenated with the field name such that a user could not edit the Sort template.

5. Improvement to Descriptive Statistics and correction to Histogram report header.

6. If a user used the dot syntax for a date and the field is a DATE type and the TIME can be entered, an "????Invalid Entry" error message would be displayed or the data would not be displayed even though there was data that should have been displayed. For example:

   GO TO FOIA REQUESTED DATE: LAST// 1.MAR.97
   (NOIS: CLA-0197-20856, VAC-1197-21817)

7. If a user was using the "!" in a print field and printed the same field again, an erroneous Count was reported which was twice the number of counted items.

   Example #1: In this example the output is correct as documentated the v22 Getting Started Manual, Section: Printing Statistics Only.

```
 Select OPTION:    PRINT FILE ENTRIES
 OUTPUT FROM WHAT FILE: NUMBER AND KIDS//
 SORT BY: NAME//
 START WITH NAME: FIRST//
 FIRST PRINT FIELD: !NAME
 THEN PRINT FIELD:
 Heading (S/C): NUMBER AND KIDS STATISTICS  Replace
 DEVICE:   Telnet terminal
 NUMBER AND KIDS STATISTICS                JUL 23,1999  13:34    PAGE 1
          NAME
 ------------------------------------------------------------------------

 COUNT     3

    Example #2: This is an example of the fixed output.  Please note
the
    PRINT FIELD entry order does not matter.

 OUTPUT FROM WHAT FILE: MUMPS OPERATING SYSTEM//
 SORT BY: NAME//  START WITH NAME: FIRST//
 FIRST PRINT FIELD: !NAME
 THEN PRINT FIELD: NAME
 THEN PRINT FIELD:
 Heading (S/C): MUMPS OPERATING SYSTEM STATISTICS  Replace
 DEVICE:   SYSTEM
 MUMPS OPERATING SYSTEM STATISTICS          SEP 14,1999  07:06    PAGE
1
          NAME                             NAME
 ------------------------------------------------------------------------

          CACHE/OpenM                      CACHE/OpenM
          DSM for OpenVMS                  DSM for OpenVMS
          DTM-PC                           DTM-PC
          GT.M(VAX)                        GT.M(VAX)
          MSM                              MSM
          OTHER                            OTHER
          -----------------------------
 COUNT    6
```

11.0 DI*22*17 SEQ #11: Miscellaneous Bug Fixes In DIC Lookup

11.1 Description

Miscellaneous bug fixes in the FileMan lookup ^DIC routines

11.1.1 Routines: DIC1, DIC2, DIC3

Support the undocumented "U" flag so that MailMan can get the same display on one of their lookups as they did in V21. This supports a lookup on the index with no transforms. I.e., the code expects the lookup value to be an internal format (of a pointer, date, etc.), and will look on the index for a match, and will not try to convert the lookup value in any way. Reported by Gary Beuschel on a local message at the San Francisco ISC.

11.1.2 Routines: DIC, DIC1

Make the "Y" flag in DIC(0) work the same as it did in version 21. This undocumented flag is supposed to make ^DIC return a list of entry numbers that match the lookup value in Y(n) where 'n' is an IEN. The FIND^DIC call is now a better way to do the same thing. Reported by Lucille Barrios on a FORUM mail message.

11.1.3 Routines: DICLIX1, DICU11

I found during my testing that in cases where the Lister decides to build a temporary index on a pointer or variable pointer because it's the fastest way to return the list in order, I needed to save the internal pointer or variable pointer on the temporary list, in case the user asked to have it returned in the output.

11.1.4 Routine: DICA

With the "E" flag, UPDATE^DIE assumes that the values passed in the FDA are in external form. Also, for LAYGO/Finding (?+) nodes, the value of the .01 field is used as a lookup value to see if an existing record should be edited, or a new record added. Prior to this change, the Updater used the internal form of the .01 as a lookup value, rather than the external form. If the .01 is a pointer or variable, this meant the lookup would fail. For DINUM'd pointers, this resulted in the Dialog error message #302, "Entry 'xxx' already exists" to be returned.

12.0 DI*22*21 SEQ #12: Editing Screened Laygo Pointers

12.1 Description

This patch fixes a problem that occurs when you edit a pointer field that has a screen and allows LAYGO into the pointed-to file. If the variable D happens to be set when the input transform of the pointer field is executed and isn't changed by the input transform, ^DIE is passing it to MIX^DIC1 to do the lookup on the pointed-to file, even though D isn't a valid list of index names.

The problem can manifest itself as bad variable name errors at S+2^DIQ:1 or null subscript errors at N5+16^DICN0 when the DEBTOR field (#9) of the ACCOUNTS RECEIVABLE file (#430) is edited.

13.0 DI*22*22 SEQ #13: New-Style Indexes, X1,X2 Arrays

13.1 Description

This patch fixes the following problem: If a new-style cross-reference is defined on a field, ^DIE or FileMan's 'ENTER OR EDIT FILE ENTRIES' is used to edit a multiple field and the field with the new-style cross-reference, then the X1 array (which contains the old values of the fields in the cross-references) may be set incorrectly after the user descends into and back out of the multiple.

For example, suppose MFIELD is a multiple field, a new-style cross-reference contains field MFIELD1, and you edit the fields as follows:

```
Select MFIELD: ENTRY 1// <RET>
  MFIELD1: TEST 1// <RET>
Select MFIELD: <RET>
FIELD1: first// second
```

When the new-style cross reference on FIELD1 is executed, X1 and X1(1) contain 'second' instead of 'first'.

14.0 DI*22*23 SEQ #14: Subscript Error Routine DIO4

14.1 Description

14.1.1 Bug Fixes

14.1.1.1 Routine: DIO0

1. An error of <SUBSCRIPT>HDR^DIO4:3 would occur if the following conditions were true:

A. After Sorting there were No Records To Be Printed.

and

B. If the Header was a Print Template that contained Fields that were to be printed.

14.1.2 Routine: DIP3

2. During SQA it was also found that if you answer:

HEADING: TEST LIST // [<sp>

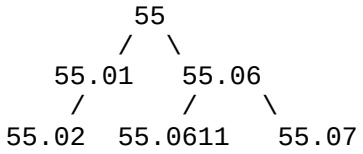
where you are telling FileMan to use the space-bar retrieval of the Template previously named, FileMan would use the "[" as the header.

15.0 DI*22*27 SEQ #15: Cross Reference Compiler, DA Array

15.1 Description

This patch fixes a problem with compiled cross-references. If a file has its cross references compiled and an IX^DIK, IX1^DIK, or ^DIK call is made to reindex or delete an entry in a multiple, and that multiple in turn has multiples underneath it, it was possible to get an undefined DA(n) error.

This bug was found when an entry in a multiple in file #55 was reindexed with IX^DIK. Part of the file structure tree for file #55 is:



If the cross-references on file #55 are compiled under version 22 of FileMan, and IX^DIK is used to reindex an entry in subfile #55.06, it was possible to get an undefined DA(2) error, when the cross-references in subfile #55.0611 were executed.

16.0 DI*22*24 SEQ #16: Undefined Routine: DIDX, Line: B+1

16.1 Description

16.1.1 Bug Fix

If a user asked for a BRIEF Data Dictionary Listing and "^" at the following prompt:

```
Select LISTING FORMAT: STANDARD// BRIEF
ALPHABETICALLY BY LABEL? No// ^
```

and then jumped to an option who's Type was set to Inquiry, an Undefined type error would be reported.

Routine: DID

17.0 DI*22*25 SEQ #17: Additional Print Fixes

17.1 Description

17.1.1 Bug Fix

It was reported that when a FileMan valid internal date/time was being used as a TO value for EN1^DIP the date/time was being rejected as Invalid.

Routine: DIP1

It was reported when using a long Customized Header, for example, THE PRINT FIELD: <my field>;"PAYMENT DATE TO RECONCILE TIMESPAN" not all of the Customized Header would print.

Routine: DIL

18.0 DI*22*20 SEQ #18: VP Screens, Long .01, DIC(P)

18.1 Description

Fixes several bugs in the FileMan lookup ^DIC routines and the finder FIND^DIC.

18.1.1 Routines: DIC1, DIC2, DIC5

When the "T" flag in DIC(0) is used, FileMan displays the field on which a match is made, as it presents the entries. This was happening even when DIC(0) did not contain "E". This patch prevents the display unless DIC(0) contains "E". (Found by Gary Beuschel, MailMan developer).

18.1.2 Routines: DIC3, DICF, DICF2, DICL2

A .01 field that was 240 characters long caused a SUBSCRIPT error to occur in the Finder FIND^DIC calls. I also found a similar problem in ^DIC if the lookup value was very long. This patch corrects both problems. (Reported by Hellevi, Ruonamaa of Finland.)

18.1.3 Routines: DICM3

When doing a lookup on an indexed variable pointer field where the pointer had a screen, the screen logic wasn't being honored during lookup on the pointed-to file. Found by Frank Stalling and Paul Sharp.

18.1.4 Routines: DICM0

If a New Style index on the .01 field has a TRANSFORM FOR STORAGE on it, then don't allow LAYGO to the file while the value is in it's format after going through the transform, as it may not be proper internal storage format. An example is a date transformed to backwards collating sequence, 9999999-2991214. Found by developer.

18.1.5 Routines: DIC, DIC0

The new feature that makes ^DIC set DIC("P") so that it doesn't have to be set as an input variable by the developer when allowing LAYGO to multiples, wasn't working correctly. Found by Jerald F Rutherford.

18.1.6 Routines: DIC1, DIC2, DICM, DICM1

In the process of testing the Name Standardization project for the New Person file, it was discovered that when there was a transform on the "B" index, the index value wasn't being displayed correctly during lookup.

18.1.7 Routines: DIC4

When the new "T" flag was used in DIC(0), developer discovered that a hard error could occur if one of the indexes was a new style index with TRANSFORM FOR STORAGE code. The error happened during the code that executes the transform against the lookup value, then does a second lookup on the index looking for a match to the transformed lookup value.

18.1.8 Routines: DIC3

Entries were not being found doing lookup when DIC(0) contained the undocumented flag "U", as it is when doing a relational jump using a backwards pointer.

18.1.9 Routine: DICUIX

Look for the new field 'TRANSFORM FOR LOOKUP' in the INDEX file. Previously during lookup, if FileMan failed to find a match to the lookup value, and if a lookup index had TRANSFORM FOR STORAGE code, FileMan executed that against the lookup value and tried to find a match with the new transformed value. Now it uses the TRANSFORM FOR LOOKUP code instead. The TRANSFORM FOR STORAGE code is used only to transform an entries field value before storing it in the index for that entry.

18.1.10 Routines: DIC3, DIC5, DICM

Because of the place where the DICR array was being NEWed when DIC(0) contained the new "T" flag, I was losing a flag that told me I asked the user whether they wanted to add a new entry while I had an internal, transformed lookup value to stuff in the .01 field. This was causing me to ask again whether they wanted to add a new entry, after all the transforms were done. Found by developer.

18.1.11 Routine: DIKCFORM

When a new-style index is created and a TRANSFORM FOR STORAGE code was defined on one of the cross-reference values, the TRANSFORM FOR DISPLAY field was made required. This change makes the TRANSFORM FOR DISPLAY optional, even if a TRANSFORM FOR STORAGE is defined, since the internal form of the index value may already be readable to the user.

18.1.12 Routines: DINIT2A3, DINIT2A4, DINIT0FA, DINIT0FB

Add a new field to the Index file (#.11), Cross-References Values multiple (#.114). The field is TRANSFORM FOR LOOKUP (#5.3) and is used by FileMan's lookup utilities (see change to routine DICUIX above). Note that this new field is transported as part of the KIDS build for this patch, so there is no need to re-run DINIT to get the new field installed.

18.1.13 Routines: DINIT0FA, DINIT0FB

When a new-style cross-reference is edited, provide a default for Subscript Number only for Regular-type indexes, not MUMPS-type indexes. Also, set the Subscript Length to 30 for Free Text and MUMPS fields only if the the field is used as a subscript. Note that these changes to the ScreenMan form for editing new-style cross-references is transported as part of the KIDS build for this patch, so there is no need to re-run DINIT to get these changes.

18.1.14 Routine: DIKCP1

Include the new field TRANSFORM FOR LOOKUP (#5.3) in Data Dictionary Listings.

19.0 DI*22*29 SEQ #19: Undef On Lookup To 200, User Enters Spaces

19.1 Description

Fixes UNDEF error on ^DIC call (Classic FileMan lookup).

19.1.1 Routines: DICM1

Fxes an UNDEF error when user enters a lookup value that results in X being returned as null from TRANSFORM FOR LOOKUP code on a new style index. Found in Clarksburg, WV during testing of the Name Standardization for New Person file patch, which introduces a TRANSFORM FOR LOOKUP on the "B" index of file 200. The bug happened only when the user entered a lookup value of either 3 or more spaces, or punctuation.

20.0 DI*22*8 SEQ #20: Screenman/WP Fixes

20.1 Description

20.1.1 Routine: DDS0

Kill the DDSSTACK variable at the end of the DDS call. DDSSTACK can be set within a ScreenMan form, and if not killed, could confuse subsequent calls to ^DDS.

20.1.2 Routines: DDS01, DDS02, DDSM1

Allow a form-only field to be used as the selection field of a repeating block.

20.1.3 Routine: DDSVALF

Allow the DA array to be passed into $$GET^DDSVALF and PUT^DDSVALF via the IENS parameter.

20.1.4 Routines: DDS5, DDSM

When a lookup is done into a subfile at the Select prompt of a multiple field, or at the selection field for a repeating block, check the value of the new fields ASK 'OK' for the multiple or the repeating block. If the value is 'YES', ScreenMan will ask "OK?" if the lookup value matches exactly one entry in the subfile.

20.1.5 Routine: DDS41

If the user attempts to save a Form and a required form-only field is not filled in, and if the Caption of the form-only field is null, use the Unique Name in the error message to the user. Before this change, ScreenMan would print something like: "Page x, is a required field."

20.1.6 Routines: DINIT293, DINIT294, DINTI295, DINIT297

These DINIT routines contain the two new fields:

Subfile #.4032, Field #10: ASK 'OK'

Subfile #.4044, Field #6.6: ASK 'OK'

Since the KIDS build for this patch brings in these fields automatically, there is no need to rerun DINIT after installing this patch. (See the above changes to Routines DDS5 and DDSM for a description of how these new fields are used.)

20.1.7 Routines: DINIT0F2, DINIT0FF, DINIT0FG, DINIT0FI

These DINIT routines include the new fields

Subfile #.4032, Field #10: ASK 'OK'

Subfile #.4044, Field #6.6: ASK 'OK'

in

Form DDGF BLOCK EDIT (#.40301), Block DDGF BLOCK EDIT OTHER (#.403013)

and

Form DDGF FIELD DD (#.40403), Block DDGF FIELD DD OTHER MULTIPLE (#.404033)

which are used to edit the properties of multiple fields and repeating blocks on a ScreenMan form. Since the KIDS build for this patch brings in these two forms automatically, there is no need to re-run DINIT after installing this patch. (See the above changes to Routines DDS5 and DDSM for a description of how these new fields are used.)

20.1.8 Routines: DINIT00T, DINIT00U

These DINIT routines correspond to the Dialog file entries #9211-9214 brought in by this patch. These four Dialog file entries are the help screens for the Screen Editor, and have been edited to reflect the key sequence changes introduced with FileMan V. 22:

1. The <Home> and <End> keys on PC's, or the <Find> and <Select> keys on VT220's, can be used to move the cursor to the beginning and end of the text on the current line.

2. The <PF1>Q keys sequence quits the editing session, as before, but if there are any edits, the Screen Editor prompts: Do you want to save changes? The help screen has been edited to say that <PF1>Q is used to "Quit with optional save."

3. The <Insert> key on PC's, or the <Insert Here> key on VT220's, can be used to toggle insert/replace mode.

4. To search for a string of text, only the <PF1>F key sequence can be used. The <Home> or <Find> key can no longer be used to search for text, since they now move the cursor to the beginning of the text (see 1 above).

20.1.9 Routines: DDSWP, DDW, DIWE

If in the screen-mode version of MODIFY FILE ATTRIBUTES, you edited the Description of the field with the Screen Editor, an undefined DDGLVID error would occur.

20.1.10 Routines: DICATTD, DICATTDK, DINIT0F8

If a user creates a new field via the screen-mode version of MODIFY FILE ATTRIBUTES, and then immediately tries to delete the new field without giving the field a data type, the user cannot exit the form with <PF1>E.

20.1.11 Routines: DINIT0F4, DINIT0F8, DINIT0FF

Change the behavior of the form invoked by the screen-mode version of MODIFY FILE ATTRIBUTES: if a field is deleted, exit the form, rather than go to the Command Line.

20.1.12 Routines: DINIT0FF

If in the Utility Function menu, Edit File option, the users deletes the value of FILE NAME in order to delete the entire file, this patch changes the behavior so that the user is automatically exited from the form, rather than taken to the Command Line.

20.1.13 Routine: DICATTD

If a user edits the DESCRIPTION or TECHNICAL DESCRIPTION of a field via the screen-mode version of MODIFY FILE ATTRIBUTES, and then decides to discard the edits and presses <PF1>Q to quit the Screen Editor, the edits may still appear if the user subsequently goes back in to edit those fields. The edits were not saved, but were still in a buffer.

20.1.14 Routine: DDSZ2

If the fields on a repeating block on a ScreenMan form take up more than one row, and undefined N() error could occur when the form is compiled.

20.1.15 Routines: DIE, DDS5, DDSM

If identifiers are added to a subfile, or if a multiple field's "Are you adding" prompt is first defined as NO, then later changed to YES, the header node of existing data in multiples is not updated. Therefore, when a lookup is done on an existing multiple, the changes don't take effect. This patch changes ^DIE and ScreenMan, so that when a lookup is done on a multiple, the header node is updated, if necessary.

20.1.16 Routine: DIWE

If ^VA(200,0) exists, but is not called NEW PERSON, FileMan assumes that the NEW PERSON file does not exist, and doesn't obtain the user's preferred editor from that file. With this patch, FileMan issues the following warning:

> **WARNING**: You appear to have a file #200 stored at ^VA(200), but it is not named 'NEW PERSON.' I will assume your preferred editor is the Line Editor.

20.1.17 Routine: DIWE2

If the Line Editor is used, and a line is added that contains many tab characters, $C(9), it is possible to get a string too long error as the Line Editor tries to replace each tab character with the string "TAB" enclosed in vertical bars. With this patch, if the string starts to get too long during the replacement, the Line Editor starts replacing each tab character with a single space instead.

20.1.18 Routine: DIWE3

If the line editor is used to compose a MailMan message, and Transfer option is selected, responding with a message number at the "From what text:" prompt was not working. For example:

```
    1>
  EDIT Option: Transfer incoming text after line: 1
  From what text: 1381514  <SELECT FILE TO TRANSFER FROM>
  From what text:
```

20.1.19 Routine: DIWE3

If you use the Transfer option in the Line Editor, it was possible to select word processing fields on files to which you don't have access.

20.1.20 Routine: DIEFW

If in a call to WP^DIE, the Word Processing Filer, you pass in an array in which the lines of text are not stored on 0 nodes, and there are also non-canonic subscripts in the array, you could get an undefined error. For example:

```
  >S ^TMP($J,1)="Line 1"
  >S ^TMP($J,2)="Line 2"
  >S ^TMP($J,"ABC",1)="Non-canonic node"
  >D WP^DIE(16000,"1,",11,"","^TMP($J)")

  %DSM-E-UNDEF, undefined variable ^TMP(123456789,"ABC")
  -DSM-I-ECODE, MUMPS error code: M7
  %DSM-I-ATLABEL, PUTWP+9^DIEFW:2 . I $G(DIEFWPFL)'["Z"
  S @DIEFNODE@(LINECNT,0)=@DIEFTSRC@(INLINE)
```

21.0 DI*22*9 SEQ #21: Export Fixes

21.1 Description

21.1.1 Bug Fix:

1. When using FileMan's:

   Select OPTION: OTHER OPTIONS

   Select OTHER OPTION: DATA EXPORT TO FOREIGN FORMAT

   Select DATA EXPORT TO FOREIGN FORMAT OPTION: EXPORT DATA

   If during the dialog the user selected to Delete the template when the export was completed and then began the Search dialog but decided to abort the Search for some reason the Export template would be deleted.

2. If a user's DUZ(0) contained the "@" sign, the Read and Write Access Security for the Export template was being set to Null, instead of being set to the "@" sign.

3. Template Security was not being honored in the same manor that other FileMan Print templates, that is, a person who's DUZ(0) was not contained in the Write Access of the Export template would be allowed to delete the template.

4. During internal SQA of this patch it was discovered that an intended API for the EXPORT function had not been distributed with version 21. This patch corrects that.

21.2 Format

D
EXPORT^DDXP(FILE,EXPORT_TEMPLATE,DELETE_FLAG,SORT_TEMPLATE,[.]FR, [.]TO,.DIS,[.]DISTOP,IOP,DQTIME)

21.3 Input Parameters

FILE (Required) Is the File Number from where the data to be Exported is located. Note: A special case is when exporting data from file number 1.1, the Audit file, in this case FILE then becomes "1.1^<file number of the audited file>". For example, if the audited data is associated with the Patient file, then the string would look like:

"1.1^2"

EXPORT_TEMPLATE (Required) Is the name of the Export template, without the surrounding brackets "[]", that was created when the developer used option: CREATE EXPORT TEMPLATE.

DELETE_FLAG (Optional) Indicates whether or not the Export template should be deleted when the exporting of the data has finished. It has two possible values:

0(zero) Do NOT delete the Export template when the export has finished. Default.

1 DELETE the Export template when the export has finished.

SORT_TEMPLATE (Optional) Is the name of the Sort template, without the surrounding brackets "[]", that will be used for file Sorting. If this parameter is Null then the user will see the standard FileMan Sort dialog.

[.]FR (Optional) This is exactly equivalent to the FR input variable(s) for the Classic FileMan print DIP and is passed to the print by reference.

[.]TO (Optional) This is exactly equivalent to the TO input variables(s)for the Classic FileMan print DIP and is passed to the print by reference.

.DIS (Optional) This is exactly equivalent to the DIS array input variable(s) for the Classic FileMan print DIP and is passed to the print by reference.

[.]DISTOP (Optional) This is exactly equivalent to the DISTOP array input variable(s) for the Classic FileMan print DIP and is passed to the print by reference.

IOP (Optional) This is exactly equivalent to the IOP input variable for the Classic FileMan print DIP and is passed to the print by reference.

DQTIME (Optional) This is exactly equivalent to the DQTIME variable for Classic FileMan print DIP and is passed to the print by reference.

The following are some examples of usage. Please note that in all of the examples the DELETE_FLAG is null, i.e. 0(zero).

** Example 1 - Where no Sort template is provided and the user is asked Sort dialog:

```
>D EXPORT^DDXP(2,"ZZS0 SKIP TEST")
SORT BY: NAME//
START WITH NAME: FIRST//
DEVICE:
```

** Example 2 - Where a Sort template is provided:

```
>D EXPORT^DDXP(2,"ZZS0 SKIP TEST",,"ZZS0 TEXPORT #1")

* Previous selection: DATE ENTERED INTO FILE from Jan 1,1997 to
Jun 4,1999
START WITH DATE ENTERED INTO FILE: FIRST// 1/1/97  (JAN 01, 1997)
GO TO DATE ENTERED INTO FILE: LAST// T  (JUN 07, 1999)
DEVICE:
```

** Example 3 - Where a Sort template is provided and the FROM and TO value are supplied:

```
>S FR="1/1/97"
>S TO=DT
>D EXPORT^DDXP(2,"ZZS0 SKIP TEST",,"ZZS0 TEXPORT #1",FR,TO)
DEVICE:
```

************ Audit File (1.1) Special Case ************

Because users might want to Export information from the Audit file (1.1) a special case was created. All parameters that are to be pass remain the same as above EXCEPT for the FILE parameter. In this special case the format is a follows:

FILE "1.1^<file number of audited file>"

** Example:

```
>D EXPORT^DDXP("1.1^16200","ZZSO",,"ZZS0 AUDIT")

* Previous selection: DATE/TIME RECORDED from Jan 1,1997 to
Dec 31,1997@24:00 START WITH DATE/TIME RECORDED: FIRST// 1/1/97
 (JAN 01, 1997)
GO TO DATE/TIME RECORDED: LAST// 12/31/97  (DEC 31, 1997@24:00)
DEVICE:
```

******* Sample Sort Template, Export Template, and Routine *******

In this example what we want to do is send a brochure using Microsoft's Word, Mail Merge to the new patients who visited the Medical Center in the previous month. For purposes of illustration we are going to assume the month in question was October of 1996.

**Sort Template Used:

```
NAME: ZZSO NEW PATIENTS//
READ ACCESS: @//
WRITE ACCESS: @//
SORT BY: ]NAME//
* Previous selection: NAME not null
START WITH NAME: FIRST//
```

```
  WITHIN NAME, SORT BY: DATE ENTERED INTO FILE  Replace
   * Previous selection: DATE ENTERED INTO FILE from Oct 1,1996 to
 Oct 30,1996
  START WITH DATE ENTERED INTO FILE: FIRST// 1/1/96  (JAN 01, 1996)
  GO TO DATE ENTERED INTO FILE: LAST// 1/1/96  (JAN 01, 1996)
    WITHIN DATE ENTERED INTO FILE, SORT BY:
 STORE IN 'SORT' TEMPLATE:    ZZSO NEW PATIENTS
          (Jun 17, 1999@05:14) User #9152 File #2    SORT

 DATA ALREADY STORED THERE....OK TO PURGE? NO// YES
 DESCRIPTION:
  1>Get previous month's New Patients for mass marketing mailing.
 EDIT Option:

 SHOULD TEMPLATE USER BE ASKED 'FROM'-'TO' RANGE FOR 'DATE ENTERED INTO
 FILE'? NO// YES

**Export Template Used:
 NAME: ZZSO PATIENT ADDRESS X
  DATE CREATED: JUN 17, 1999@08:26
  READ ACCESS: @                          FILE: PATIENT
  USER #: 9152                            WRITE ACCESS: @
  DATE LAST USED: JUN 17, 1999            TEMPLATE TYPE: EXPORT
 FIELD ORDER: 1                            DATA TYPE: FREE TEXT
 FIELD ORDER: 2                            DATA TYPE: FREE TEXT
 FIELD ORDER: 3                            DATA TYPE: FREE TEXT
 FIELD ORDER: 4                            DATA TYPE: FREE TEXT
 FIELD ORDER: 5                            DATA TYPE: FREE TEXT
  EXPORT FORMAT: EXCEL (COMMA)            SUB-HEADER SUPPRESSED: YES
 HEADER (c): @@
 FIRST PRINT FIELD: W $C(34)//
 THEN PRINT FIELD: NAME;X//
 THEN PRINT FIELD: W $C(34);X//
 THEN PRINT FIELD: W $C(44);X//
 THEN PRINT FIELD: W $C(34);X//
 THEN PRINT FIELD: STREET ADDRESS [LINE 1];X//
 THEN PRINT FIELD: W $C(34);X//
 THEN PRINT FIELD: W $C(44);X//
 THEN PRINT FIELD: W $C(34);X//
 THEN PRINT FIELD: CITY;X//
 THEN PRINT FIELD: W $C(34);X//
 THEN PRINT FIELD: W $C(44);X//
 THEN PRINT FIELD: W $C(34);X//
 THEN PRINT FIELD: STATE;X//
 THEN PRINT FIELD: W $C(34);X//
 THEN PRINT FIELD: W $C(44);X//
 THEN PRINT FIELD: W $C(34);X//
 THEN PRINT FIELD: ZIP CODE;X//
 THEN PRINT FIELD: W $C(34);X//
 THEN PRINT FIELD: W $C(44);X//
 COMPILED (c): NO
```

**Example routine and output:

```
 ZZSONPAD --
        ;SFISC/SO-Sample Export API Usage ;7:18 AM  17 Jun 1999
       ;;1.0
       N %DT
       S %DT="AEPX"
       S %DT("A")="Enter Beginning of previous Month: "
       D ^%DT
       I Y<1 Q
       S FR=","_$P(Y,".")
       S %DT="AEPX"
       S %DT("A")="Enter End of previous Month: "
       D ^%DT
       I Y<1 Q
       S TO=","_$P(Y,".")
       K %DT
       D EXPORT^DDXP(2,"ZZSO PATIENT ADDRESS X",,"ZZSO NEW
PATIENTS",FR,T
        O)
       Q FM22 >D ^ZZSONPAD

Enter Beginning of previous Month: 10/1/96  (OCT 01, 1996) Enter End of
previous Month: 10/30/96  (OCT 30, 1996) DEVICE:   Telnet terminal
"BIRD,TWEETY","123 TREE ST.","SAN FRANCISCO","CALIFORNIA","94521",
"BUNNY,BUGS","123 CARROT ST","SAN FRANCISCO","CALIFORNIA","90041",
"CAT,SYLVESTER","132 ANY ST","SAN FRANCISCO","CALIFORNIA","98765",
"DUCK,DAFFY","301 Howard St.","San Francisco","CALIFORNIA","94105",
"LASVEGAS,LEACH","111 LAS VEGAS BLVD.","LAS VEGAS","NEVADA","89101",
"RUNNER,ROAD","234 ROAD ST.","SAN FRANCISCO","CALIFORNIA","94077",
"SAM,YOSEMITE","234 YOSEMITE","SAN DIEGO","CALIFORNIA","98765",
```

22.0 DI*22*26 SEQ #22: Modify File Attributes Fixes

22.1 Description

22.1.1 Bug Fixes

1. Using the Modify File Attributes option, if a user steps into a Variable Pointer field and answers "Yes" to the question:

   SHOULD ENTRIES BE SCREENED:

   and does any one of the following:

   a.) Hit Return at the prompt to enter the MUMPS code for the SCREEN.
   b.) "^" exit at the EXPLANATION OF SCREEN prompt.

   this leaves the "Yes", that entries should be Screened, but no Screen code, thus causing an Undefined error. Routine DIEV1 will now check for this condition and continue as if there was No Screen applied to the lookup.

   Routine: DIEV1

   Routine DICATT4 has been modified such that when the user answers "Yes" to the question: "SHOULD ENTRIES BE SCREENED," they must answer the next 2 prompts:

   SCREEN
   EXPLANATION OF SCREEN

   If the user has made a mistake in answering "Yes" to: SHOULD ENTRIES BE SCREENED they can escape by doing the following:

```
SHOULD ENTRIES BE SCREENED: NO// Y  YES
SCREEN: ^SHOULD ENTRIES BE SCREENED  <=note the use of ^SHOULD
SHOULD ENTRIES BE SCREENED: YES// N  NO
```

   Note: Routine DIPR26 sets the second piece of ^DD(.12,1.0) to RFX to make SCREEN a Required field and routine DIMIT11A was changed so that if a site needs to reinstall FileMan this change will not be lost.

2. It was reported by the development community that when entering a Screen for a Pointer To A File Data Type, the code was not being checked to see if it contained DIC("S"). Not having the code containing DIC("S") is valid but the developer was not being warned in any way that the code being entered might

have some undesirable results. This has been fixed so that you will be warned if the code does not contain DIC("S"). For example:

```
SHOULD 'NEW PERSON' ENTRIES BE SCREENED? No// Y  (Yes)
MUMPS CODE THAT WILL SET 'DIC("S")': D ^MYPROG
        WARNING - Screen Does Not Contain DIC("S")
EXPLANATION OF SCREEN:
 Routine: DICATT5
```

3. The Traditional Cross Reference help for the Cross-Reference Number has been expanded upon such that it now reads:

   CROSS-REFERENCE NUMBER: 1// ?

You may use the number shown if you are the custodian of the file this cross-reference is in. If you are not the custodian of the file, you should select a number that corresponds with a numberspace for which you have custody. Questions regarding numberspace custody may be referred to: DBA@FORUM.VA.GOV

23.0 DI*22*30 SEQ #23: Reader (DIR) Changes for MailMan

23.1 Description

23.1.1 Bug Fix

1. As reported in the nois: NYN-0599-11112, a person could enter a Reversed number sequence when deleting VistA email messages, for example, 220-26. This would be interpreted as meaning messages 26-220, when the user could have just as easily meant 220-260. This patch fixes that particular problem in that strict sequencing for a range of numbers will be followed. For example, 220-26 will not be allowed, but 220-260 will be allowed.

   This patch also fixes several other deficiencies that were discovered during internal SQA:

   a.) The default number of decimals is 0. It was discovered that a number with decimals would be allowed.

   b.) Y and Y(0) were being returned with partial data even though the user's input contained invalid data.

   c.) If the developer specified a range of values within the DIR(0) the upper bound of that range was not being honored.

   d.) Under certain conditions if an error was found within the user's input, the error code was not being passed back to the calling program for processing.

   Routine: DIR3

2. The MailMan developer requested that a new parameter be created so that the Replace/With dialog could be suppressed. For example:

   Send mail to: <name over 19 characters long> Replace

   After patch XM*7.1*118 it will look like:

   Send mail to: <name over 19 characters long>//

   We have added a new parameter to the 1st piece of DIR(0) - "r" - that means suppress the Replace/With dialog and use the standard "//".

   Note: Full implementation of this change by MailMan will not occur until the release of MailMan patch XM*7.1*118.

Routine: DIR

3. If a Maximum date was not supplied to the Reader, the user would be prompted with a Maximum date of 99/99/2699, this has been fix so the user will now see: 12/31/2699.

   Routine: DIR2

23.2 Documentation Changes

Input Variable: DIR(0)
Read Type: 'Free-Text'
'Pointer'
'DD'
PIECE-1; Subsequent Characters (optional);
New Character - "r"

If user does not choose to accept the default response, they must type in their entire response. They will not get the 'Replace-With' prompt, no matter how long the default response is.

24.0 DI*22*32 SEQ #24: Non-displaying 'n MATCH(ES) FOUND' Message

24.1 Description

24.1.1 Bug Fixes

A problem was discovered with FileMan's Search option that when the following conditions were true the trailer message 'nn MATCH(ES) FOUND' was not being printed:

a.) A Search was done.

b.) A multi-level Sort was used.

c.) And NO Print fields were selected.

The Search would find the records and store them in a Search if one was selected, but the user had no knowledge of whether the errored out, ran out of spool space and etc.

Routine: DIO2, DIO4, DIP5

25.0 DI*22*28 SEQ #25: Fixes To DIC Lookup And Lister Call

25.1 Description

Fixes bugs in the FileMan lookup ^DIC routines, and the Lister LIST^DIC routine.

25.1.1 Routines: DICUIX1

Improve speed of question mark help in the classic FileMan lookup. Specifically the help took a long time to return when processing an IFCAP file that pointed to a very large file. Reported by Deborah Ernzen.

25.1.2 Routines: DICUIX2

When doing a LIST^DIC call using a whole-file index, an error message was generated when trying to return the external value from the index, when the ctual indexed field was down in a multiple. Reported by user outside VA.

25.1.3 Routines: DIALOG

Fixes an 'illegal number' error that could happen when this code restores the naked reference, and the last subscript of the naked reference looks like a number in scientific notation (ex., 1E-114 TRIAGE CLINIC). Found by Roy Gaber at Bay Pines.

25.1.4 Routines: DICF0

An UNDEF error was found and fixed by the FileMan developer. This is a highly unlikely chain of events. The error is caused when a) The developer calls FIND^DIC passing the name of a compound index to the call, and b) there is no lookup value passed for the first subscript on the index. Then c) The Finder selects an alternate index to use for the initial loop through the file but d) that alternate index has no Subscript fields set on the Cross Reference Values multiple.

25.1.5 Routines: DIC3, DICUIX, DICUIX2

A SUBSCRIPT error was found and fixed by the FileMan developer. This error occurred when a very long lookup value was entered in a call to ^DIC. The fix also improves the code used to find an entry when the lookup value is longer than the subscript value in the index.

26.0 DI*22*33 SEQ #26: Modify File Attributes ScreenMan Version

26.1 Description

26.1.1 Bug Fixes

If a user in the Screen Editor form of "Modify File Attributes" to create a new field and the Data Type is POINTER, the user can not use the File Name to make the selection. They are forced into using the File Number. This patch corrects that problem.

Routine: DINIT0F6 (sent as a courtesy, so if a site has to reinstall FileMan, the edits to Form DICATT will not be lost.)

Form: DICATT

27.0 DI*22*36 SEQ #27: CPRS/LAB Requested Enhancement to FileMan Date/Time

27.1 Description

27.1.1 Enhancement

An Enhancement was requested by the CPRS/LAB development such that the FileMan date/time abbreviation "NOW" would recognize an addition/subtraction of minutes. Currently the "NOW" abbreviation only recognizes plus/minus "W"eek, "M"onth, "D"ay, "Y"ear.

Routine: DIDTC/%DTC

27.2 Documentation Changes

Since the letter 'M' or 'm' designates to FileMan "Month", it was decided that the mathematical abbreviation for minutes, the apostrophe, would be used to indicate the plus/minus value of minutes. For example, if you wanted to enter a date and time 6 minutes in the Future from the current date and time, you would enter: NOW+6' On the other hand, if you wanted to enter a date and time 8 in the Past from the current date and time, you would enter: NOW-8'. The following table illustrates acceptable abbreviations when entering time:

Abbreviation	**Meaning**
NOW+3'	Present time plus 3 minutes
NOW+1H	Present time plus one hour.
NOW+3D	Present time plus three days.
NOW+4M	Present time plus four months.
NOON	Today at 12:00 noon.
MID	Today at 24:00 midnight.

28.0 DI*22*39 SEQ #28: Refresh (<PF1>R) In ScreenMan Form

28.1 Description

28.1.1 Routine: DDS01

This patch corrects a bug introduced with patch DI*22*8. If in a ScreenMan form, the user presses <PF1>R to refresh the screen, ScreenMan goes into an infinite loop.

29.0 DI*22*11 SEQ #29: Triggers And Compound Indexes

29.1 Description

This patch fixes problems associated with how triggers interact with new-style cross references introduced in FileMan Version 22.0.

29.1.1 Routines: DDS02, DDS4, DIE, DIE1, DIE17, DIE2, DIED, DIEF, DIEF1, DIEZ, DIEZ1, DIEZ2, DIEZ3, DIEZ4, DIKC, DIKC1, DIKC2,

If in a ^DIE, FILE^DIE, or ^DDS call, the following sequence of events occurred:

1. Field A, which has a record-level index, is edited (or filed).
2. Field B is edited (or filed), and a trigger on Field B changes the value of Field A.
3. The record-level index on Field A is fired at the end of the editing session.

Problem fixed: The record-level index set in step 3 contained the value of Field A as edited in step 1, rather than the actual value as triggered in step 2.

29.1.2 Routine: DICR

Problem fixed: When trigger logic was executed, new-style cross-references on the triggered field were not getting fired.

29.1.3 Routine: DIKCUTL1

Problem fixed: Existing trigger logic checks whether cross-references exists on a triggered field, and only then, does it call ^DICR to fire those cross-reference. However, the check for the existence of cross reference didn't account for new-style cross-references that may have been created on the triggered field. To fix this problem, when the user creates a new-style cross-reference, FileMan will look at each field used in the cross-reference.

If the field is triggered, FileMan modifies the trigger logic to call ^DICR unconditionally, to ensure that new-style cross-reference are fired when the trigger logic is executed. FileMan will also recompile any input templates that contain fields that trigger fields in the new-style cross-reference.

29.1.4 Routines: DIFROMS2, DIFROMSX, DIFROMSY

Change: When a new-style cross-reference is installed for a field, check to see if the field is triggered. If so, modify the trigger logic to call ^DICR unconditionally to

ensure that new-style cross-references are fired when the trigger logic is executed. Also the 0 node of the INDEX and KEY files was not being correctly updated when bringing in a KEY or INDEX that previously existed at the target site.

29.1.5 Routine: DIKCDD

Change: Don't allow a new-style cross-reference to be created on a computed field.

29.1.6 Routine: DIKCP

Problem fixed: If the top level of a file contains no new-style cross-references, when you print a Standard DD Listing on the file, the first new-style cross-reference defined in a multiple would print out twice.

29.1.7 Routine: DIPR11 (Pre-Install Routine)

This new routine is the Pre-Install Routine for Patch DI*22*11. It looks at all fields used in new-style cross-references created since FileMan version 22.0 was installed. For each field, it checks to see whether the field is triggered, and if so, modifies the trigger logic to call ^DICR unconditionally, to ensure that the new-style cross-references are fired when the trigger logic is executed.

For each field used in new-style cross-references, DIPR11 also recompiles any input templates that contain fields that trigger the field to ensure that the compiled routines contain the modified trigger logic.

The pre-install routine also corrects the 0 node for the data on the INDEX (.11) and KEY files (.31), in case they were incorrect due to a bug in the KIDS routine that brings in KEY and INDEX entries.

> **NOTE**: You can safely delete routine DIPR11 after installing this patch.

30.0 DI*22*31 SEQ #30: Bug Fixes For DIC Lookup Routines

30.1 Description

Fixes several bugs in the ^DIC lookup and FIND^DIC Finder routines as listed below.

30.1.1 Routines: DIC1, DIC5, DICF4, DICM, DICM0, DICM2, DICN

DIC("W") variable caused 'String too long' error. FileMan builds code into DIC("W") to display the identifiers. If the .01 field is a pointer, it also builds identifiers for all the pointed-to files. This local variable could get too long with a long pointer chain. The patch corrects this by building overflow code in the DIC("W",n) array for sequential integers 'n'. Problem found by Greg Cebelinski and Frank Traxler.

30.1.2 Routines: DICF

UNDEF error at CLOSE+6^DICF. If a null value, or a value containing control characters was passed in the VALUE input parameter, and if user did not pass a TARGET ROOT parameter, then an UNDEF error happened at CLOSE+6^DICF.

30.1.3 Routines: DICN0,DIC2

UNDEF error at F1+3^DIC3. Change NEWs DIC and other variables to protect them before executing the new-style indexes on the .01 field of a new entry. A bug was caused by the fact that a new-style index changed the value of DIC. It was fired on a new entry, then later the FileMan code was checking for the existence of the new entry, using DIC for the global reference. Caused an UNDEF error. Reported by Patrick Redington.

30.1.4 Routines: DIDU

Fixes 'bad variable name' error when an invalid internal variable pointer value is passed to the $$EXTERNAL^DILFD call. This was found in Tucson when values like "12;" were passed to the call. A valid internal variable pointer value in their case would have looked like "12;DPT(".

30.1.5 Routines: DICN

The 3rd piece of the 0 node of file ^PSRX is sometimes getting reset to a small number. This could happen if the input variable DINUM was left hanging around before a call to ^DIC or FILE^DICN. I found one place where that variable could be left behind after a call to FILE^DICN, this patch makes sure it gets killed.

31.0 DI*22*6 SEQ #31: DIM/DICOMP Fixes

31.1 Patch Description

31.1.1 Bug Fixes

Note - because in general no one routine was involved with a particular fix and because of the overlapping of fixes, routines for a specific fix have not been listed.

1. When a Variable-Pointer field was used in a Computed expression a MaxString error would occur.

2. A Null SubScript error would occur when using a BackWards Pointer within a Computed expression.

3. Sometimes when creating a Computed field expression, a MaxString error would occur. Note: this error is different than the MaxString in #1.

4. When printing from a multiple and you choose the name of the multiple, even though the .01 field name of the multiple is different, that choice would be allowed and an endless loop would occur.

5. When Queueing a Search, it was possible to get a MaxString error.

6. When using a Backward Pointer during the Search, the user was not being asked the: "Do You Mean..." question.

7. During the Transferring of entries to a Higher file number, a MaxString error could occur.

8. When Searching on a field that is a Pointer to another file and the user does not have Read Access to the Pointed to file, the request is rejected. This is inconsistent with the Sort and Print where the user is granted Read Access to the .01 field of the Pointed to file. This patch corrects that inconsistency.

9. When entering M code into a MUMPS type field, the following example would be rejected:

   S:$G(ORTYPE)'="Z" Y=$S($$INPT^ORCD:9,1:15)

   This patch corrects the problem.

10. When sorting on a field who's name contained an arithmetic operator, generally the "/", and there was another field within that file that starts with the word(s) before the arithmetic operator, then DICOMP would think that the user meant for the arithmetic operator to be applied.

11. When using a Extended pointer within an Edit template the code generated for DIC("S") was invalid "M" code.

12. When sorting on a field and using the "M" Contains the sort would fail to find the correct records.

13. When sort on a multiple that contains a full relational jump, for example, MY FIELD:THEIR FILE:THEIR FIELD, an Undefined error would occur.

32.0 DI*22*34 SEQ #32: Print - Time Fixes

32.1 Description

32.1.1 Bug Fixes

When queuing a FileMan report and at the prompt:

"REQUESTED TIME TO PRINT: NOW//"

if the user requested their report to be queued using for example "NOW+1H", this syntax it would be rejected.

Routine: DIP4

During internal SQA it was discovered that if a user was sorting by a field who's type was a DATE/TIME. And the user entered the time down to the Seconds, the sort would include records who's time was Less Than the specified beginning time.

Routine: DIP1

33.0 DI*22*19 SEQ #33: DIQ Fixes

33.1 Patch Description

33.1.1 Bug Fixes

33.1.1.1 Routine: DIQ1

If a developer when using the API call EN^DIQ1 attempts to use a global array to store the output, then the local symbol table will be killed off. The use of a global array is INVALID when using the Classic API EN^DIQ1, the output can only be stored into a local array.

But on the other hand, FileMan should not be killing off the local symbol table. This patch corrects the problem of filing off the local symbol table.

33.1.1.2 Routines: DIQ; DIDC

If a user selects a "Customed-Taliored", "List File Attributes" and sorts on a Label or NUMBER that is also contained in a Multiple, then an UNDEF will occur. This patch corrects the UNDEF problem.

34.0 DI*22*37 SEQ #34: Fix Trigger Logic Code/ Undefined DIV Error

34.1 Description

34.1.1 Routine: DICE4

This patch corrects a bug introduced in patch DI*22*6. If a trigger-type cross-reference is created with conditional logic, and the code that FileMan generates must be stored on more than one node in the ^DD global, an undefined DIV error can occur when the trigger logic is executed.

For example, if you create a Trigger-type cross reference on the .01 field of a file to trigger a date-type field, and the Set Logic value is 'TODAY' and the Set Condition is INTERNAL(DATE)<2990101', when a record is added to the file, an undefined DIV error would occur.

35.0 DI*22*5 SEQ #35: NEWing of the Variable DINUM

35.1 Patch Description

Notice: See "Installation Instructions" section for Installation Warning.

35.1.1 Bug Fix

Near the end of testing of FileMan v22, Patch DG*5.3*149 was release to the field. This patch caused the NOIS Reporting Site to be unable to Register patients. The DG routine involved relied on the "leaking" variable DINUM. Because it was going to be some time before PIMS could patch their code, because of higher priority issues, it was felt by the FileMan team that a temporary "work around" patch would be issued to allow the "leaking" of the variable DINUM until such time the PIMS team could patch their code. The "work around" was incorporated in patch DI*22*1. The PIMS team has now patched their code, DG*5.3*245 SEQ #245, Released: 4/12/00. This patch is being issued to remove the temporary "work around" that was incorporated in FileMan patch DI*22*1.

36.0 DI*22*35 SEQ #36: Illegal Number at %+3, Routine %DT

36.1 Description

36.1.1 Bug Fix

If a user enters a numeric style date at a Date/Time prompt and for some reason a bunch of numbers are entered, for example, a key gets stuck, then %DT will error out at %+3 with an Illegal number. This patch will prevent the Illegal number and ask the user to reEnter the date/time.

Routine: DIDT/%DT

37.0 DI*22*3 SEQ #37: Enhancement, Bug Fix To Help For Lookup

37.1 Patch Description

** Notice: See "Installation Instructions" section for Installation Warning **

Enhancement to online "?" help for lookup, allows user to designate index to be used, 'start with' values, and 'partial match' values when displaying the list of entries in the file during online help. Also fixes bugs that caused parts of the online help to scroll off the screen.

37.1.1 Routines: DDSU, DIEQ

If user types "??" when asked to enter data in a field, they are shown the DESCRIPTION for that field. If the description is too long, it scrolls off the screen. NOIS: ISF-0399-60937, by Gary Beuschel

37.1.2 Routines: DDSU, DICQ, DICQ1

Fixes various problems that were causing list of entries displayed during online help to scroll off the screen. Found by developer.

37.1.3 Routines: DICQ, DICQ1, DICUIX1

Support new input variables to ^DIC and DQ^DICQ calls. These will be used during question-mark help in classic ^DIC lookups. They are used as parameters to the Lister call used to display the current entries on the file. See documentation for the new input variables following. E3R #5127 and #13948.

37.1.4 Routines: DICQ

When a ^DIC lookup is done with more than one index, and some of those indexes index more than one field, the online help implies that you will be prompted for all indexed fields. In fact this happens only if you do a lookup ONLY on the compound index alone. This patch fixes the online help to more accurately reflect that if the first lookup index is on a single field, then only the first subscript on compound indexes will be checked for a match. Found by Patrick Redington, OIFO-Salt Lake City.

37.1.5 Routines: DICLIX

Fixes a rare problem where all entries might not be displayed during question-mark help, if using an index that collates backwards. Problem was found by the developer.

37.1.6 Routines: DICLIX, DICL3

Fixes a couple of problems in doing Lister calls with compound indexes, both found by developer. First one caused Lister call to not return all entries when following a pointer chain more than 2 files deep. The other one caused some entries to not be returned when following a pointer chain where one of the files in the chain had no "B" index.

37.2 Documentation Changes

Documentation for the new features will appear in the VA FileMan Programmers manual at http://vista.med.va.gov/infrastructure/documentation.html#FileMan.

37.2.1 Classic FileMan, Routine Call ^DIC

DIC("?PARAM",file#,"INDEX")=Index name. (Optional). Used to control entries displayed during online "?" help only. If provided, this index will be used to display the entries from the file specified by file#. Otherwise, FileMan uses the first lookup index specified for the ^DIC call. This value is used as the INDEX parameter to the Lister call to display the entries. See documentation for LIST^DIC for more information.

DIC("?PARAM",file#,"FROM",n)=value. (Optional). Used to control entries displayed during online "?" help only. This array can be set to define a starting value for an entry in the lookup index used to list entries from the file. Integer value 'n' is associated with the 'nth' subscript in the index (ex. regular old-style indexes always have just one indexed data value so 'n' would be 1). If a starting value is defined for subscript 'n', then starting values must also be defined for all of the subscripts preceding 'n'.

This information is used to set the FROM parameter for a call to LIST^DIC in order to display the entries in the file specified by file#. Therefore the entries must meet the same rules as the FROM parameter described in that call. See documentation for LIST^DIC for detailed information.

If DIC(0) contains an "L" and the indexed field is a pointer, then after displaying the current entries on the file, FileMan allows the user to see entries on the pointed-to file. In that case, the developer may request starting values for any pointed-to file in the pointer chain.

If the user enters "^value" when asked whether they wish to see the entries in the file, the value entered by the user will override the starting list value passed by the developer in this array.

DIC("?PARAM",file#,"PART",n)=value. (Optional). Used to control entries displayed during online "?" help only. This array can be set to define partial match value(s) for each of the 'n' subscripts on the lookup index used during online help. The information is used to set the PART parameter for a Lister call to display the

entries. See documentation for LIST^DIC for more information. This is similar to DIC("?PARAM",file#,"FROM",n) described above, but defines a partial match value rather than just a starting value.

37.2.2 Classic Calls, Rutine DQ^DICQ

DIC("?N",file#)=n (Optional) Use this variable in the same way it is described as input to ^DIC.

DIC("?PARAM",file#,"INDEX")=index name. (Optional) Use this input array in the same way it is described as input to ^DIC.

DIC("?PARAM",file#,"FROM",n)=value. (Optional) Use this input array in the same way it is described as input to ^DIC.

DIC("?PARAM",file#,"PART",n)=value. (Optional) Use this input array in the same way it is described as input to ^DIC.

37.2.3 Classic FileMan, Routine Call ^DIC: Examples

B) File 662002 has a .01 field that points to the NEW PERSON file (#200). In this example, we'll use input arrays in DIC("?PARAM",662002,"FROM",1) to start the list of entries in the "B" index of file 662002 with the letter "M". Since DIC(0) contains "L" (user can add entries to the pointed-to file 200), FileMan will also display entries from file 200, so we use DIC("?PARAM",200,"PART",1) to display only entries that start with the letter "S".

```
>S DIC=^DIZ(662002,DIC(0)="AEQZL"
>S DIC("?PARAM",200,"PART",1)="S"
>S DIC("?PARAM",662002,"FROM",1)="M"

>D ^DIC

Select ZZTAMI POINT TO NEW PERSON PERSON NAME: ??

   Choose from:
   MANNERS,JULIA      NOV 11, 1961     ANOTHER GREAT PROGRAMMER     JM
 PROGRAMMER
   MARSHALL,DELBERT MAY 05, 1965     WIZARD     TOAD          PROGRAMMER
   OGDEN,MARSHALL JUL 07, 1977     GREAT PROGRAMMER     MO PROGRAMMER
   RETROMAN,USER K M JR      JAN 01, 1969     COOLDUDE     UR
   TIMOTHY,S J      APR 03, 1948     KOOL KAT     SJT PROGRAMMER
   WERLY,BIRD      JUN 12, 1955     GROOVY GUY     BW PROGRAMMER
   WINNER,BIG      AUG 28, 1949     COMPUTER SPECIALIST     BW
 PROGRAMMER
   WINNER,SMALL      AUG 28, 1948     SECOND PLACE     SW PROGRAMMER
   ZERO,BOB      MAR 02, 1948     VERY GOOD PROGRAMMER     BZ
 IRMFO     PROGRAMMER

        You may enter a new ZZTAMI POINT TO NEW PERSON, if you wish
```

```
   Choose from:
   SHARED,MAIL
   SMITH,JOHN HOWARD STEVEN II
   STRALL,SEG        SAS
   SUPERMAN,JOE X Y JR
```

C) Using the same files as in example "B", we will display entries from the pointing file 662002, using the "AC" index, which sorts the entries by TITLE, then by NAME. In this case, we will limit the number of entries displayed at one time from both file 662002 and file 200 to 5.

```
 >S DIC="^DIZ(662002,",DIC(0)="AEQZL"
 >S DIC("?PARAM",662002,"INDEX")="AC"
 >S DIC("?N",662002)=5
 >S DIC("?N",200)=5

 >D ^DIC

 Select ZZTAMI POINT TO NEW PERSON PERSON NAME: ??

    Choose from:
    A STATE   CALIFORNIA,MR      MAR 01, 1875     A STATE     MC     ABCD
    ANOTHER GREAT PROGRAMMER   MANNING,DARYL      NOV 11, 1961     ANOTHER
GREA
 T PROGRAMMER     DM          PROGRAMMER
    BROKER WHIZ   CROSS,BOB      FEB 05, 1950     BROKER WHIZ     BC
    COMPUTER SPECIALIST   WILLY,TED     AUG 28, 1949 COMPUTER SPECIALIST TW
    COOLDUDE   RETROMAN,USER K M JR      JAN 01, 1969     COOLDUDE     UR

                ^

      You may enter a new ZZTAMI POINT TO NEW PERSON, if you wish

  Answer with NEW PERSON NAME
  Do you want the entire NEW PERSON List? Y  (Yes)
 Choose from:
    ATESTMAN,BOB K III        BKA
    CALIFORNIA,MR        MC     ABCD
    CLARK,KENT        KC
    CROSS,BOB        BC          PROGRAMMER
    DELANCY,NAN        ND          PROGRAMMER
```

38.0 DI*22*40 SEQ #38: Lookup Value >200, Lookup On New Person File

38.1 Description

Allow lookup value up to 250 characters.

38.1.1 Routines: DIC11

Lookup value in interactive mode was limited to 200 characters to avoid SUBSCRIPT errors in the code that $O through the lookup indexes. For files with a .01 field longer than 200 characters, this did not allow users to add new records to the file in interactive mode. Code has been fixed to avoid the errors in another way, so the lookup value can now be up to 250 characters long.

38.1.2 Routines: DICM

User did a lookup on a file whose .01 field was a variable pointer. When the lookup resulted in selecting an entry on file 200 (NEW PERSON), one of the pointed-to files, and when the lookup value was entered in lower case, the NAME field from the NEW PERSON file didn't display. When lookup value was entered in upper case, it did.

38.1.3 Routines: DIC2, DIC3

When lookup index was on a variable pointer, and DIC("PTRIX") input array was used to do lookup on a pointed-to file using something other than the "B" index, part of the lookup index value wasn't being displayed. Found by developer.

39.0 DI*22*48 SEQ #39: File~DICN Creates Incorrect Global, Error in Y+24~DIC1

39.1 Description

Fixes three errors. The first one is a STRLEN (string too long) hard error at Y+24^DIC1. The second happens in a call to FILE^DICN. When user passes an invalid global reference, it can cause an invalid global node to be created. The third was a null subscript error in MODIFY FILE ATTRIBUTES, when ? help was requested at the DESTINATION field prompt.

39.1.1 Routines: DIC1

Fileman presented a list of matches to the user's lookup value. Instead of selecting one of the numbered entries, the user entered a very long string of characters. Caused a %DSM-E-STRLEN error (string too long) at Y+24^DIC1.

39.1.2 Routines: DICN0, DILIBF

Invalid global reference DIC="^DIC(16," was passed to a FILE^DICN call. Instead of failing, FileMan added the requested entry, and created a node ^DIC(16,0)="^^1^1".

39.1.3 Routines: DILIBF, DIDU

The DESTINATION and GROUP fields on file 0 are not completely FileMan compatible. They sort of look like multiple pointers. The V22 question mark help used on those two fields from MODIFY FILE ATTRIBUTES now calls LIST^DIC to display the current entries, as well as the entries available for selection. Special code needed to be added to account for the fact that the DD structure for these fields is quite normal.

40.0 DI*22*51 SEQ #40: Bad Syntax Error At S2+7~DICL2

40.1 Description:

Error "bad syntax at end of command or Xecute string" occurred in FORUM.

40.1.1 Routines: DICF1

Error "bad syntax at end of command or Xecute string" occurred in FORUM when user entered an invalid value containing quotes at the SEND TO prompt. Error happened during a $$FIND1^DIC lookup to the NEW PERSON file (200), when a match was made between the first comma piece of the lookup value, and the Nickname value of a person.

41.0 DI*22*38 SEQ #41: Fix Bogus Export Message When Queuing

41.1 Description

41.1.1 Bug Fix

When a user used the 'EXPORT DATA' option and answered "NO" to the question:

> "Do you want to delete the <template name> template after the data export is complete?"

and then queued the output. The user would receive the following bogus message:

> "Export template <template name> will be deleted when queued export is completed."

This patch corrects this problem.

> **Routine**: DDXP4

42.0 DI*22*43 SEQ #42: Incorrect Display When Editing a Print Template

42.1 Description

42.1.1 Bug Fix

If a user was editing a Print Template in the traditional Roll and Scroll mode and they encountered a Computed field that was followed by another Computed field that had a prefix, for example the '!', the display of the first Computed field would be corrupted and the user could do no further editing of the template.

Routine: DIP22

43.0 DI*22*44 SEQ #43: Corrupted Var. 'I' & Field Level Access

43.1 Description

43.1.1 Bug Fixes

1. If a user selected a field who's Data Dictionary type was 'Computed' and the computation involved a Pointer to another file, one of the two following problems would occur:

   A. Only the first record would print.

   B. Or a never ending loop printing the first record.

   This patch corrects this problem.

   Routine: DICOMP1

2. If a user was sorting on a field that Pointed to another file, a screen, DIC("S"), was being set to test if the user had Read access at the Field level. If the fields Read access node was set, but it was set to Null then the test would fail. This patch corrects that problem. Also, if you have any questions related to how Field level security is enforced, please review the "Data Security" section, Part III, Chapter 12, of the FileMan User's Manual.

   Routine: DICOMPY, DICATTD8

44.0 DI*22*45 SEQ #44: Missing Header and Trailer When No Records

44.1 Description

44.1.1 Bug Fix

When a user was using a Kernel Print type menu option, that is where FileMan went into 'silent' mode and there were no records found for the report. Instead of printing the Report Header and the message "No Records to Print", FileMan would return control back to the menu prompt. This patch corrects that problem.

Routine: DIO4

45.0 DI*22*47 SEQ #45: UNDEF Variable 'Y' @EN+1~DIFGO

45.1 Description

45.1.1 Bug Fix

If a FileMan user uses one of the following Filegram options from the Kernel menu system:

- Create/Edit Filegram Template [DIFG CREATE]
- Display Filegram Template [DIFG DISPLAY]
- Generate Filegram [DIFG GENERATE]
- View Filegram [DIFG VIEW]
- Specifiers [DIFG SPECIFIERS]
- Install/Verify Filegram [DIFG INSTALL]

and then uses another Filegram option while still at the Filegram options prompt, an UNDEF error will occur. This patch corrects this problem.

Routine: DIFGO

46.0 DI*22*50 SEQ #46: <UNDEF>@UP+7 Routine: DITP

46.1 Description

46.1.1 Bug Fix

If a programmer used the entry point P^DI and Deleted an entry and then asked FileMan to "DELETE ALL SUCH POINTERS" an Undefined would occur at UP+7. This patch corrects this problem.

Routine: DITP

47.0 DI*22*46 SEQ #47: Null WP Nodes Loses Formatting

47.1 Description

47.1.1 Bug Fix

When a user was editing a Word Processing field and entered a <cr> for line spacing, certain Word Processing editors would set the global node equal to a null. FileMan was expecting the node to set to a space <sp> when using the Classic Programmer call: D^DIWP

Routine: DIWP

48.0 DI*22*54 SEQ #48: Executable Help That Calls DIC

48.1 Description

48.1.1 Bug Fix

If a user enters a question mark '?' for help And there is executable help on that field And if within that executable help a call is made to DIC, then an Undefined at: C4+11^DICUIX2 will occur. This patch fixes that bug.

Routine: DDSU, DICQ1

49.0 DI*22*56 SEQ #49: FILE~DICN, Multiple's & DIC('P')

49.1 Description

49.1.1 Bug Fix

If a call was made to FILE^DICN and a Multiple was being set that had no entries and DIC("P") was not set then the new entry would not be set.

This patch corrects this problem.

Routine: DICN0

50.0 DI*22*53 SEQ #50: Pre-Install File Deletion & Security Codes

50.1 Description

50.1.1 Bug Fix

Currently the DIFROMServer only allows File Protection Security to be brought in If And Only If a file is new at the target site. The problem is if the developer needs to delete and reinstall the file at the target site, File Protection codes will not be brought in and are erased. The reason is KIDS during the dialog portion, checks to see if the file currently exists on the target's system and if file already exists, KIDS then flags the file as not a new file and passes that bit of information to the DIFROMServer. Then when the install begins and the Pre-Install deletes the file so as to make it new, a recheck by the DIFROMServer is not made on the target's system, thus causing the File Protection codes not to be brought in. This patch now corrects that situation and the DIROMServer will now check again to see if the file still exists on the target system and if file Does Not exist, it will then consider the file as new and bring in the File Protection Security codes to the target's system.

Routine: DIFROMS2

50.1.2 Enhancement

To allow developers to just change the File Security Codes at a target site without having to transport the entire file, a supported Data Base Server (DBS) API has been created to allow developers to set the various File Security Codes.

Routine: DDMOD

50.1.3 Documentation Changes

Database Server (DBS) API

FILESEC^DDMOD(): Set File Protection Security Codes

This entry point sets the security access codes for a file, which are stored in the following nodes:

^DIC(filenumber,0,"AUDIT") -- Audit Access
^DIC(filenumber,0,"DD") -- Data Dictionary Access
^DIC(filenumber,0,"DEL") -- Delete Access
^DIC(filenumber,0,"LAYGO") -- LAYGO Access
^DIC(filenumber,0,"RD") -- Read Access
^DIC(filenumber,0,"WR") -- Write Access

50.1.3.1 FORMAT

FILESEC^DDMOD(FILE,.SECURITY_CODES,MSG_ROOT)

50.1.3.2 Input Parameters

FILE (Required) File number. (Cannot be less than 2.)

.SECURITY_CODES (Required) Array of new security access codes:

SECURITY_CODES("AUDIT") = Audit Access
SECURITY_CODES("DD") = Data Dictionary Access
SECURITY_CODES("DEL") = Delete Access
SECURITY_CODES("LAYGO") = LAYGO Access
SECURITY_CODES("RD") = Read Access
SECURITY_CODES("WR") = Write Access

MSG_ROOT (Optional) The root of an array into which error messages are returned. If this parameter is not included, errors are returned in the default array: ^TMP("DIERR",$J).

50.1.3.3 OUTPUT

None

50.1.3.4 Examples

Example 1
In this example we are going to set all of the File Security Code nodes:

```
D ^%G
 Global ^DIC(16028
         DIC(16028
 ^DIC(16028,0) = ZPATR FILE^16028
 ^DIC(16028,0,"GL") = ^DIZ(16028,
 ^DIC(16028,"%",0) = ^1.005^^0
 Global ^

 S SECURITY("DD")="@"
 S SECURITY("RD")=""
 S SECURITY("WR")="A"
 S SECURITY("DEL")="@"
 S SECURITY("LAYGO")="@"
 S SECURITY("AUDIT")="@"
 D FILESEC^DDMOD(16028,.SECURITY)

 D ^%G
 Global ^DIC(16028
         DIC(16028
 ^DIC(16028,0) = ZPATR FILE^16028
 ^DIC(16028,0,"AUDIT") = @
```

```
^DIC(16028,0,"DD") = @
^DIC(16028,0,"DEL") = @
^DIC(16028,0,"GL") = ^DIZ(16028,
^DIC(16028,0,"LAYGO") = @
^DIC(16028,0,"RD") =
^DIC(16028,0,"WR") = A
^DIC(16028,"%",0) = ^1.005^^0
```

Example 2:
In this example we are going to use the results from the previous example and change just the Write Access:

```
S SECURITY("WR")="a"
D FILESEC^DDMOD(16028,.SECURITY)
D ^%G

Global ^DIC(16028
        DIC(16028
^DIC(16028,0) = ZPATR FILE^16028
^DIC(16028,0,"AUDIT") = @
^DIC(16028,0,"DD") = @
^DIC(16028,0,"DEL") = @
^DIC(16028,0,"GL") = ^DIZ(16028,
^DIC(16028,0,"LAYGO") = @
^DIC(16028,0,"RD") =
^DIC(16028,0,"WR") = a
^DIC(16028,"%",0) = ^1.005^^0
Global ^
```

50.1.3.5 Error Codes Returned

401 File does not exist or the File Number that was passed was Less Than 2.

51.0 DI*22*57 SEQ #51: '??' Help and eXecutable Help

51.1 Description

51.1.1 Bug Fix

If a user was doing '??' help on a field AND that field has the eXecutable Help defined AND the definition for the field is a Screened Pointer AND from within the eXecutable Help a call is made to ^DIC, THEN a double list of possible selections would be displayed. This is because within the Input Transform for the field, a call is also made to ^DIC.

For example:

```
 The DD defination: ^DD(<file #>,<field #>,4) = D HELP^MYPROG
 Routine: MYPROG
 HELP     S X="??",DIC="^VA(200,",DIC(0)="EQ",D="B" D IX^DIC Q

 The result displayed:
NAME: FIRST TEST//
Select PT200 MULTI: ??

     You may enter a new PT200 MULTI, if you wish


   Choose from:
   <first name>        MC     ABCD
   <second name>        KC
   <third name>        DPC          PROGRAMMER
   <and so on>

   Choose from:
   <first name repeated>        MC     ABCD
   <second name repeated>        KC
   <third name repeated>        DPC          PROGRAMMER
   <and so on repeated>
```

Routine: DICQ

52.0 DI*22*61 SEQ #52: Histogram and Y2K

52.1 Description

52.1.1 Bug Fixes

When a user sorted and triggered Statistics on a field that is defined as just a Date or used the DATE function on a field that is defined as a Date/Time And then used FileMan's Histogram option, for any date Greater than the year 1999, the year would be displayed as 100. A 4-digit year will now be displayed.

Routine: DIH

53.0 DI*22*55 SEQ #53: Fields Validator And Primary Key

53.1 Description

53.1.1 Routines: DIEV, DIEVS

Bug fixed: If you make a VALS^DIE call and pass the data for fields in a Primary Key, and you pass that data in Finding or LAYGO/Finding nodes in the FDA. VALS^DIE may pass back an error message indicating that the new values are invalid because they would create a duplicate key, even though you intend to use the values for look up, rather than for filing.

This patch allows you to pass a new "K" flag to VALS^DIE to indicate that you wish to use the Primary Key fields, rather than the .01 field, for lookup in Finding and LAYGO/Finding nodes. This flag is equivalent to the "K" flag in the Updater (UPDATE^DIE).

54.0 DI*22*58 SEQ #54: Automatic Reindexing Of Regular Indexes

54.1 Description

54.1.1 Routines: DICD, DICE, DIKCUTL3

Prior to this patch, if you created a traditional or new-style REGULAR index, FileMan would automatically loop through all the entries in the file and build the index for you.

This could present a problem if there are many entries in the file. The looping could take a long time and would be unnecessary if for all the entries in the file, there is no data in the field(s) being indexed.

This patch modifies the Utility option for creating, editing, and deleting cross-references so that it asks you whether you want to build the new and delete the old index.

55.0 DI*22*60 SEQ #55: Jump '?' When Editing Display

55.1 Description

55.1.1 Bug Fix

When a user was editing a file using FileMan and the user entered at a field prompt "^?" to see what fields could be jumped to, the user would see the proper field numbers for the file, but Not the proper field Labels.

Routine: DIE0

56.0 DI*22*62 SEQ #56: Unable to Enter 8 Character Routine Name in 'Special Lookup'

56.1 Description

56.1.1 Bug Fixes

When a user was using the FileMan option: Utility Functions/Edit File AND they wanted to enter a routine in the following field: LOOK-UP PROGRAM: only a maximum of 6 characters where allowed, instead of 8. The DIEDIT Form has been corrected to allow for 8 characters.

Routine: DINIT0FE & DINIT0FF are being send as a courtesy and DINIT does NOT need to be rerun.

57.0 DI*22*65 SEQ #57: $$GET1~DID Does Not Return Values

57.1 Description

57.1.1 Bug Fixes

During a routine Data Base Intergration Agreement (DBIA) review it was discovered that when using the DBS call $$GET1^DID And requesting the value for "HELP-PROMPT", "XECUTABLE HELP", DELETE ACCESS, READ ACCESS, or WRITE ACCESS, a 202 error was being falsely returned.

Routine: DIQGDD

58.0 DI*22*69 SEQ #58: Audit Date Stored in External Format

58.1 Description

58.1.1 Bug Fix

All of the data stored in the Audit file (#1.1), for fields:

2.1 OLD INTERNAL VALUE

3.1 NEW INTERNAL VALUE

is being stored in internal format except for Date/Time data, which was being stored in external format. This patch corrects this situation.

Routine: DIET

59.0 DI*22*63 SEQ #59: Transfer Option Corrupts DD

59.1 Description

59.1.1 Bug Fix

If a user was using the FileMan TRANSFER option And they choose a File Number that Contains the Old File Number, Then the Data Dictionary for the new file will be corrupted.

Routine: DIT1

60.0 DI*22*64 SEQ #60: Print/Caption Fixes

60.1 Description

60.1.1 Bug Fixes

1. If a user was doing a Sort And that user made a partial selection at the 'Sort By' prompt And that user Timed Out, Then an Undefined would occur at LEVELS^DIP. This patch corrects this problem.

   Routine: DIP

2. During internal SQA it was discovered that if a user selected the Captioned template for the output at the 'Print Field' prompt, And they selected a printer margin greater than 80, for example 132, the report would still print with margins of 80. This patch corrects this problem.

   Routine: DII; DIQ; DIQ1; DIWW

3. If a person was using the option 'List File Attributes' and selected the 'Custom-Tailored' style And sorted by the attribute 'Date Field Last Edited', they would have to use a FileMan internal date structure. With this patch an external date representation can now be used.

   Routine: DIP1

61.0 DI*22*74 SEQ #61: Fix Date Returned By D of DIQ

61.1 Description

61.1.1 Bug Fixes

This patch corrects two problems introduced by patch DI*22*64 when using the supported entry point D. The date in external format was being returned without a space between the comma "," and the millennium digit. This patch restores that space. It also restores the leading zero when the day is a single digit.

Routine: DIQ

62.0 DI*22*67 SEQ #62: Lookups With Keys And New-Style Indexes

62.1 Description

62.1.1 Routine: DICN1

62.1.1.1 Bugs fixed:

1. If on a file you define a database Key that contains only the .01 field, and the file also contains identifiers, it was possible to create an entry with a duplicate .01 field.

2. If you try to add a record to a file that has a required identifier, but you don't have WRITE access to that field, you would get an undefined error at ZAP+1^DICN1.

62.1.2 Routines: DICUIX, DICUIX2

Bug fixed: Suppose you have a new-style index in which the second (or subsequent) subscript has a Transform for Display. If you perform an IX^DIC call using that index, and enter ?? at the Select ... prompt, FileMan was not executing that Transform for Display.

62.1.3 Routine: DIC11

If you do a lookup on a simple new-style index (an index that includes only one field) that is not the "B" index, FileMan's prompt would be:

'Select FileName: '.

With this patch, FileMan includes the Field Label, or the Lookup Prompt in the case of new-style indexes, in the 'Select' prompt, so that the prompt becomes:

Select FileName FieldLabel: or

Select FileName XRefValueLookupPrompt:

63.0 DI*22*81 SEQ #63: Captioned Output Scrolling Off Screen

63.1 Description

63.1.1 Bug Fixes

If a user was using FileMan's 'INQUIRE TO FILE ENTRIES' or a developer called FileMan's Classic call EN^DIQ and the IO variables located in global XUTL("XQ",$J,...) were not defined, then the first page of the output would scroll off the top of the screen before a pause was issued.

This patch corrects that situation.

Routine: DIQ

If a user was using FileMan's 'INQUIRE TO FILE ENTRIES' and selected several entries from a file, Captioned Output would scroll the display off the screen. This patch corrects that situation.

Routine: DII

If a Search was being used and the Captioned template was being used to display at the FIELDS prompt then a control 'C' was need to stop/<esc> the display. For example:

```
Search the Hospital Location file
 Criteria 'A' NAME[<something>
 Criteria 'B' APPOINTMENT (multiple)
              PATIENT (multiple)
              PATIENT[<something>
IF: A+B

<take the defaults>

FIRST PRINT FIELD: [CAPTIONED
```

Routine: DIQ

64.0 DI*22*72 SEQ #64: UPDATER FIXES

64.1 Description

64.1.1 Routine: DIDU2

When you use the Updater to add an entry to a file, and the header node of that file lacks a 3rd piece, the Updater tries to determine the correct value of the entire header node. In the process, the Updater has the following problems:

- If the name of the file contains an "A", the Updater thinks that the 2nd piece of the header node should contain an "A". This patch makes the Updater put an "A" in the 2nd piece only if the 2nd piece already contains an "A".

- To determine the value of the 4th piece, the Updater performs a manual count of all the entries in the file. If the file is very large, the count could take a long time, and since the header node is locked during this time, other processes are unable to add entries to the file. This patch makes the Updater stop counting after 10,000.

64.1.2 Routine: DIEV1

CHK^DIE and VAL^DIE rejected the external value of a variable pointer field if it was exact, but partially matched another value in the same file. For example, if both "MILK" and "MILK CHOCOLATE" are valid variable pointer values, the value "MILK" would always be rejected. This patch makes CHK^DIE and VAL^DIE accept an exact match, even if it partially matches another value in the same file. (Note that this patch corrects the situation in which there is an exact and partial match in the same pointed-to file, and no partial match in any other pointed-to file. If there is a partial match in another pointed-to file, the exact lookup value is rejected.)

This change also affects the Filer (FILE^DIE) and the Updater (UPDATE^DIE) when the "E" flag is used (i.e, external values are passed). The Updater and Filer will now accept external variable pointer values that are exact, but partially match another value in the same file.

Another related change to CHK^DIE, VAL^DIE, FILE^DIE, and UPDATE^DIE, is if the external value of a variable pointer field is passed, and that value is not in the "B" index in the pointed-to file, the value would be rejected, even if the value is in another index.

65.0 DI*22*80 SEQ #65: Release Of VA FileMan Version 22.0 Key And Index Tutorial

65.1 Description

VA FileMan Version 22.0 introduced Key and Index functionality, which provides VISTA application developers the tools to design and develop more robust, efficient, and maintainable code and files. Native support for keys and complex indexes means developers no longer need to simulate these structures with M code.

This hands-on tutorial will teach you how to take advantage of these powerful new tools. The instructions in the tutorial will guide you in creating your own keys and indexes on a test file set up via the A6AKIT set of routines.

This tutorial provides pop-up windows for additional information relevant to each lesson. Simply click on the hyperlinks and the windows will pop-up on your screen.

All test data used in this tutorial (e.g., names, Social Security Numbers [SSN], dates of birth [DOB], etc.) is fictitious and is used solely for the purpose of illustrating lesson topics.

65.2 FORMATTING CONVENTIONS

Several formatting methods are used to highlight different aspects of this tutorial:

1. "Snapshots" of computer online displays (i.e., roll-and-scroll screen captures/dialogues) and computer source code are shown in a non-proportional font.

2. User's responses to online prompts will be in boldface type.

3. Boldface type is also used to highlight a descriptive word or sentence.

4. The Enter (or Return) key is illustrated by the symbol <Enter> when displayed in computer dialogue and is included in examples only when it may be unclear to the reader that such a keystroke must be entered.

5. All uppercase is reserved for the representation of M code, and field and file names (e.g., the SSN field).

65.3 PRESENTATION STRUCTURE

65.3.1 Environment Setup

The exercises in this tutorial make use of a simple test file referred to as ZZINDIVIDUAL that contains some sample data. The A6AKIT set of routines can

be used to install this test file on an M (MUMPS) system. Environment Setup contains information on how to obtain the A6AKIT routines. Lesson 1, Exercise 1.1. Create Your Test File shows you how to run the A6AKIT routine to install the test file.

65.3.2 Introduction

The tutorial introduction contains an overview of some of the basic key and index concepts.

65.3.3 Tutorial Lessons

Each section of this tutorial (Parts 1 through 3, shown below) contains topics and subsequent lessons. The information is presented in the following topic categories:

Part 1 - Regular Indexes includes the following five lessons. A brief description is given for each.

Lesson 1 - You will create a New-style compound index based on the date of birth (DOB) and Social Security Number (SSN) fields in your ZZINDIVIDUAL test file.

Lesson 2 - You will print the definition of your New-style compound index to view the data.

Lesson 3 - You will see how your lookup index affects the behavior of the lookup utility IX^DIC. You will also change the collation and lookup prompt for the DOB field in your "C" index.

Lesson 4 - You will use the transform properties located in ScreenMan to perform transforms on subscripts.

Lesson 5 - You will create a whole-file index based on the fields in the EMAIL multiple of the ZZINDIVIDUAL test file. You will also illustrate a use for computed cross-reference values.

Part 2 - MUMPS Indexes includes the following three lessons. A brief description is given for each.

Lesson 6 - You will learn about New-style MUMPS cross-references. It covers two topics: (1) the X, X1, and X2 arrays and (2) the set and kill logic and conditions of cross-references. You will create a New-style MUMPS cross-reference. Afterwards, you will check the behavior of your MUMPS cross-reference.

Lesson 7 - You will learn when VA FileMan executes the set and kill logic of New-style cross-references. You will also create a MUMPS cross-reference that

makes use of the VA FileMan X, X1, and X2 arrays and test your new MUMPS cross-reference.

Lesson 8 - You will use the ACTIVITY property to suppress cross- reference execution.

Part 3 - Keys includes the following three lessons. A brief description is given for each.

Lesson 9 - This lesson covers (1) Key Integrity, (2) the Uniqueness Index, and (3) Primary and Secondary Keys. You will create a key on the ZZINDIVIDUAL file.

Lesson 10 - You will print the definition of the key you created in Lesson 9 on the ZZINDIVIDUAL file with key fields NAME field (#.01) and SSN field (#.02). You will look at the Uniqueness Index automatically created and explore different methods for defining the fields in the key.

Lesson 11 - You will see examples of how VA FileMan enforces the integrity of the key you created in Lesson 9, and worked with in Lesson 10.

65.3.4 Tutorial Quizzes

Each lesson in this tutorial is followed by a quiz. Each quiz is based on the lesson preceding it. They have been designed not only for you to use as a tool to test yourself on what you learned, but to offer a summary of the lesson. The quiz scores are not being recorded anywhere. They are for your benefit only.

(Text versions of the quizzes are also available. They can be accessed via links at the top and bottom of each quiz page.)

65.3.5 Standard Data Dictionary Listing of the Test File

This section contains a standard data dictionary listing of the test file. In this listing, the name of the test file is ZZINDIVIDUAL, with file number 662nnn, stored in global root ^DIZ(662nnn).

65.3.6 Entries in the Tutorial Test File

This section contains a captioned printout, including record numbers, of the data in the ZZINDIVIDUAL test file installed by routine A6AKIT.

66.0 DI*22*52 SEQ #66: Clean Up PT, IX, TRB, and 12 Nodes In Modify File Attributes

66.1 Description

66.1.1 Routine: DICATT22

The change to this routine fixes the following problems:

- If you create a variable pointer field (say #3), and then indicate that the field is a multiple, the "PT" nodes created on the pointed-to files corresponding to field #3 were not killed. Only the "PT" nodes of field #.01 of the multiple should be present.

- If you create a pointer field (say #4), define a screen on that pointer, and then define the field as multiple, then the screen definition nodes

  ^DD(file#,field#,12) = Explanation of screen
  ^DD(file#,field#,12.1) = Screen code

  would be set for the multiple field #4, instead of field #.01 of the multiple. (The input transform of field #.01 of the multiple already sets DIC("S") properly as defined by the user.)

66.1.2 Routine: DICATT4

The change to this routine fixes the following problems:

- When a file or subfile was deleted using the documented call EN^DIU2, if the file/subfile contained pointer or variable pointer fields, the "PT" nodes on the pointed-to files were not cleaned up.

- When a subfile at least 2 levels down from the top file is deleted, if the subfile or any subfiles within that subfile had any whole-file cross-references, the "IX" nodes set at upper file levels where the cross-reference resided were not getting cleaned up.

- When a subfile is deleted, if the subfile or any subfiles within that multiple had fields with trigger cross-references that triggered fields in an upper level file level or in other files, the 5 nodes on those fields were not getting cleaned up. Also, if the field being triggered was itself in a subfile, the "TRB" nodes at files levels above that subfile were not getting cleaned up.

NOTE: This patch does not include an automatic clean-up of all stray nodes that may exist on files on the system; those nodes are

not known to actually cause any problems. If desired, you can use FileMan's Data Dictionary Utilities -> Check/Fix DD Structure to clean up most of the stray nodes from the DD for selected files.

67.0 DI*22*79 SEQ #67: Lower Case 'n' Not Recognized When Sorting Date/Time Field

67.1 Description

67.1.1 Bug Fix

If a user tried to use a lower case 'n', meaning NOW, in the 'GO TO...' prompt when doing a sort on a Date/Time field, FileMan would not properly recognize the lower case 'n', thus causing an 'Invalid Entry'.

Routine: DIP1

68.0 DI*22*42 SEQ #68: Modify File Attributes SCREEN-MODE

68.1 Description

68.1.1 Bug Fixes

If the user is the Screen-Mode version of 'Modify File Attributes' AND changed the field type, for example, from a Free Text to a Pointer, THEN the Second Piece is corrupted and data cannot be entered.

In this case a user is NOT allowed to create a Multiple when the field type is a Variable Pointer.

The following steps detail the problem:

1. Create a new field
2. The first time in the data type field, enter N for numeric

   a. The low,upper limit window pops open

   b. Do not enter anything except <ENTER> at each field
3. When back to the main window, use the arrow keys to get down to the COMMAND prompt. This was the only way I could navigate to the top ofthe form again.
4. Using arrow keys go back to top of form to the FIELD LABEL
5. Press <ENTER> to go to DATA TYPE field
6. Change field type to FREE TEXT

   a. Free text parameter pop up window comes up

   b. I entered 6 characters and pattern matched X?6N
7. Then get down to COMMAND prompt and choose exit to save
8. I got the following messages:

   THE DATA COULD NOT BE FILED.
   Page 2.2, INCLUSIVE LOWER BOUND is a required field
   Page 2.2, INCLUSIVE UPPER BOUND is a required field

When creating a multiple and choosing Subfile and Subscript no warning was issued if you were about to choose an already used Subscript.

When creating a Subfile number of decimals allowed was only 4, this has been increased to 9.

69.0 DI*22*75 SEQ #69: String Too Long When Using Screen-Mode Modify File Attributes

69.1 Description

69.1.1 Routine: DDSMSG

In the screen-mode MODIFY FILE ATTRIBUTES, it was possible to get a "string too long" error at SETDDH+2^DDSMSG, if in the course of the editing session, the value of $X gets very large.

One method of duplicating the problem is as follows:

1. Use screen-mode MODIFY FILE ATTRIBUTES to create a new field called VAR PTR":

```
Select OPTION: MODIFY FILE ATTRIBUTES
DO YOU WANT TO USE THE SCREEN-MODE VERSION? Yes// <RET>  (Yes)

MODIFY WHAT FILE: xxx

Select FIELD: VAR PTR
  Are you adding 'VAR PTR' as a new FIELD (the 9TH)? No// Y  (Yes)
   FIELD NUMBER: nnn// <RET>
```

2. Tab to DATA TYPE... and type V (for variable pointer), then <Enter>.
3. Choose a file, and enter 1 for Order. (Be sure to press <Enter> after entering the number.)
4. Don't enter anything in the pop-up window. Press <PF1>C to close it.
5. Choose another file, and enter an Order number for it. Again, press <PF1>C to close the pop-up window.
6. Delete all the order numbers.
7. Enter '^' to go to the Command line. Press <Enter> to close the page.
8. Enter '^' again to go to the Command Line.
9. Enter 'S' to Save.

70.0 DI*22*85 SEQ #70: Entering ?? In A Date Read With The 'M' Flag

70.1 Description

70.1.1 Routines: DIEFU, DIEH1, DINIT00O

Background: The DT^DILF call converts a user-supplied value into VA FileMan's internal date format and (optionally) into the standard FileMan external, readable date format. The first input parameter to DT^DILF is FLAGS, which generally contains the same flags as the %DT input variable to the ^%DT call.

The ^%DT call was modified in patch DI*22*14 to accept a new flag "M" to allow input of only month and year (not day).

Bug fixed: DT^DILF returns an error if it is passed the "M" flag:

```
>D DT^DILF("M","1/01",.RES,"","MSG")
>ZW MSG,DIERR
DIERR=1^1
MSG("DIERR")=1^1
MSG("DIERR",1)=301
MSG("DIERR",1,"PARAM",0)=1
MSG("DIERR",1,"PARAM",1)=M
MSG("DIERR",1,"TEXT",1)=The passed flag(s) 'M' are unknown or
  inconsistent.
MSG("DIERR","E",301,1)=
```

This patch makes the DT^DILF call accept the "M" flag.

In addition, if "?" is passed in the second input parameter to DT^DILF, this patch makes sure that the help text returned is correct when the "M" flag is used. (The help text for the "M" flag is sent in this patch via Dialog file entry #9110.7.)

```
>D DT^DILF("M","?",.RES,"","MSG")
>ZW MSG,DIHELP
DIHELP=8
MSG("DIHELP")=8
MSG("DIHELP",1)=Examples of Valid Dates:
MSG("DIHELP",2)=  JAN 1957 or JAN 57 or 0157
MSG("DIHELP",3)=  T    (for this month)
MSG("DIHELP",4)=  T+3M (for 3 months in the future)
MSG("DIHELP",5)=  T-3M (for 3 months ago)
MSG("DIHELP",6)=Only month and year are accepted. You must omit the
  precise day.
MSG("DIHELP",7)=If the year is omitted, the computer uses CURRENT YEAR.
  Two digit year
MSG("DIHELP",8)=  assumes no more than 20 years in the future, or 80
```

```
    years in the past.
```

This bug also manifests itself in a reader call (^DIR) when the "M" flag is passed in the DIR(0) variable. If the user enters two question marks (??) at the reader prompt, an undefined error would occur:

```
>S DIR(0)="D^::AEM" D ^DIR
Enter a date: ??

%DSM-E-UNDEF, undefined variable DIHELP
-DSM-I-ECODE, MUMPS error code: M6
%DSM-I-ATLABEL, QUES+7^DIR:1    . F DILINE=1:1:DIHELP S
 A0=DIROOT("DIHELP",DILINE) D MSG
>
```

71.0 DI*22*88 SEQ #71: DA Array When New-Style XREF On Triggered Field Is Executed

71.1 Description

This patch corrects a problem that may occur if you define a new-style cross-reference or a key on a file, and that cross-reference or key contains a field that is triggered by a field in another file. New-style cross-references executed as a result of the trigger may create erroneous index entries.

71.1.1 Routine: DICR

1. Suppose you define a new-style cross-reference on Field A, which is triggered by Field B in another file. When you edit Field B, the trigger logic updates Field A, and the set and kill logic of the new-style cross-reference defined on Field A are executed.

   However, at the time the new-style cross-reference logic is executed, the DA array reflected the record corresponding to Field B, rather than the record corresponding to Field A, the triggered field. This patch corrects this problem so that the DA array properly reflects the record being triggered.

2. Suppose you define a Key that contains Field A, which is triggered by Field B in another file. When you edit Field B, the trigger logic tries to update Field A, but before it does so, FileMan checks that the integrity of the Key would not be violated by the update; that is, that the update wouldn't cause a duplicate key, or null key field values.

   However, when FileMan checked the integrity of the key, the DA array reflected the record corresponding to Field B, rather than the record corresponding to Field A, the triggered field. This resulted in two problems:

   a. FileMan was checking the integrity of the wrong record, and could allow the trigger to update Field A, even if the update violated key integrity.

   b. If FileMan determined that the update violated key integrity, FileMan would try to restore the triggered field to its original value, but could potentially set the field for the wrong record.

72.0 DI*22*73 SEQ #72: DIR Does Upper Case Transform For Pointer & DD Read Type

72.1 Description

72.1.1 Bug Fix

When a developer use a DIR call using either a Pointer or DD read type and the field that is being read is a Pointer to another file, a duplicate list would occur when the user would enter lower case alpha for the look up value. The duplicate list is occurring because the Reader calls DIC which when doing a lookup is failing to find a lower case match in it's first pass through the cross references. DIC will then transform the lower case input into upper case and will retry to find matches in the cross references and if it finds possible matches to the user's input will present them. If the user up arrows out of the list, the Reader will then do it's own upper case transform and recall DIC which will again find matches and represent them. When the user up arrows for the second time, the Reader will then issue the Help prompt.

Routine: DIR1

73.0 DI*22*77 SEQ #73: Search Fails On Variable Pointer Fields

73.1 Description

73.1.1 Bug Fix

While doing a FileMan Search, if a user selected a field that is defined as a Variable Pointer, the search would fail because the Search routine did not recognize that it had to handle Variable Pointers differently from regular Pointer type fields.

Routine: DIS1

74.0 DI*22*78 SEQ #74: Double List of Selections File 200 and Up-arrow

74.1 Descriptio

74.1.1 Bug Fix

This defect was discovered during internal SQA of patch DI*22*73. When the user enters a lower case lookup value on file 200 and the "B" cross reference is one of the cross reference(s) that is being searched, a double list will appear. For example:

```
FM22 >S DIC=200,DIC(0)="AEMQZ" D ^DIC

 Select NEW PERSON NAME: an
      1   ANCHORAGE,MAS       EKQ
      2   ANDERIES,BILL       BA
      3   ANDERSON,WILLIAM       WA
      4   ANDREWS,JOSEPH E       JEA
      5   ANDRIES,GEORGE H       GHA
 CHOOSE 1-5: ^
      1   ANCHORAGE,MAS       EKQ
      2   ANDERIES,BILL       BA
      3   ANDERSON,WILLIAM       WA
      4   ANDREWS,JOSEPH E       JEA
      5   ANDRIES,GEORGE H       GHA
 Press <RETURN> to see more, '^' to exit this list, OR
 CHOOSE 1-5: ^

 Select NEW PERSON NAME:
```

This is because there is a new style 'B' cross reference that has Transform (Lookup) that Upper cases the user's input and DIC finds the first set of possible matches. Since the user '^', DIC then preforms a Transform on the user's input to Upper case and looks again and again finds possible matches in the 'B' cross reference, thus the second list.

Routine: DIC, DIC1

75.0 DI*22*87 SEQ #75: Allocation Error When Vertical Bar(s) Are In Passed Parameter

75.1 Description

75.1.1 Defect Repair

In certain cases when a parameter is passed into the DIALOG DBS API's that contains vertical bars (windows) followed by a space and some text, an Allocation error(VMS)/Store(CachE) will occur. For example, a MailMan message who's Subject is: "|||This is my subject||| hello" (without the quotes).

Routine: DIALOG

76.0 DI*22*86 SEQ #76: DIC(0)['OV' Flag & Up-arrow/Time Out/'?' YES/NO Prompt

76.1 Description

76.1.1 Bug Fix

When a developer used any of the Classic DIC calls, that is ^DIC, MIX^DIC1, and IX^DIC, and included the 'O' and 'V' flag in DIC(0) and the following conditions occurred:

a. There was an exact match to the users input in the current cross-reference.

b. And if the user would either enter an Up-Arrow "^" Or Time Out Or enter a "?" at the 'OK' prompt.

Then DIC would return to the caller in Y just the IEN of the next inexact match within the current lookup cross-reference, instead of a -1. This patch corrects this problem.

Routine: DIC1,DIC3

77.0 DI*22*18 SEQ #77: Screen Editor Fixes

77.1 Description

77.1.1 Routine: DDW1

This patch removes a reference to the XMSUB variable in routine DDW1.

Background: If DIWESUB is input to EN^DIWE, the Screen Editor displays the contents of DIWESUB between the "less than" (<) and "greater than" (>) symbols at the top of the screen. However, if DIWESUB is not passed to EN^DIWE, the Screen Editor checked for and displayed the contents of XMSUB, which, prior to Mailman patch XM*7.1*50, contained the Subject of the Mailman message when a message is created, or the letter "R" concatenated with the number of the original message, when a response is edited. With patch XM*7.1*50, Mailman sets the DIWESUB input variable to EN^DIWE, so the Screen Editor no longer needs to check for XMSUB.

With this patch, DBIA 1042 can be retired.

77.1.2 Routines: DDWK, DINIT00T, Dialog entry #9212

This patch adds <Ctrl-E> as a key sequence for exiting the Screen Editor, equivalent to <PF1>Q. The help screens invoked with <PF1>H also show this new key sequence. (E3R: #9252)

77.1.3 Routine: DDWT1

If you press <PF4> or <Delete> on the last line of a document, and that last line is blank, the line is deleted and the cursor moves up one line. With this patch, the Screen Editor will also then move the cursor to the end of the line. (Previously, it was possible to hold the <PF4> or <Delete> key at the end of the document and erase the entire document.) (E3R: #9516)

77.1.4 Routine: DDW6

If a space is typed in column 75 (the column just beyond the default right margin), the Screen Editor automatically wraps the line and moves the cursor to the column 1 of the next line. If a second space is entered, the cursor moves to column 2. This is a problem if you type 2 spaces after colons or periods. The space at the beginning of the line has to be manually deleted. In addition, with this patch, spaces at the end of joined or wrapped lines are retained. (E3R: #3979)

77.1.5 Routine: DDW7

If the screen is shifted to the right because the cursor is in a column greater than 80, and marked (selected) text extends from the left screen to the current screen, if that

text is deselected (with <PF1><PF1>M, for example), the line wasn't getting painted correctly.

77.1.6 Routines: DDW, DDW1, DDWK, DDWT1, DINIT00T, DIWE

This change introduces an enhancement to the Screen Editor. The user can press <PF1><PF1>S to enter an interval in minutes in which the text should automatically be saved. Also, a new input variable DDWAUTO can be passed into EN^DIWE, set to the number of minutes that text should automatically be saved. The autosave setting takes effect for only the current editing session. (E3R: #9383)

77.1.7 Routine: DDW4

Make the <Home> key (on PCs) or <Find> key (on VT320s) and the key sequence <PF1><PF1><Left> toggle between column 1 and the first non-space characters on the line (if any).

77.1.8 Routines: DDW, DDW2, DDWK, DDWT1, DIWE, DINIT00T, DINIT00U

Create a new key sequence <PF1><PF1><Tab> to allow the user to enter the column positions in which tab stops should be set. Also introduce a new input variable to EN^DIWE that allows the programmer to pass in DDWTAB equal to the tab stop positions.

78.0 DI*22*59 SEQ #78: Allocation Errors In DIE

78.1 Description

78.1.1 Routine: DIE

1. **Bug fixed**: It is possible to get an allocation error under the following conditions:

   - VA FileMan's ENTER OR EDIT FILE ENTRIES option is used to edit all fields in a file or subfile.
   - The file contains a large number of fields.
   - None of the fields in a particular range of field numbers (say fields 1 through 500) contain cross-references, none are multiple or word processing fields, and none are audited.
   - The MUMPS $S[TORAGE] intrinsic function is not a reliable measure of the amount of symbol table space available in the partition.
   - The symbol table is too small to hold the data dictionary information for all fields in the range.

   To help guard against potential allocation errors, this patch makes DIE gather the data dictionary information for a maximum of 50 fields at a time, rather than use $S[TORAGE] to determine the available symbol table space.

78.1.2 Routine: DIED

2. **Bug fixed**: If you use FileMan's ENTER OR EDIT FILE ENTRIES option or ^DIE and use a 4-slash stuff on a multiple field, it is possible to inadvertently change the .01 field of the record in the file. For example:

```
Select OPTION: ENTER OR EDIT FILE ENTRIES

INPUT TO WHAT FILE: ZZMY MULTIPLE//
EDIT WHICH FIELD: ALL// 100////xyz <Enter>
   EDIT WHICH MULT 100 SUB-FIELD: ALL//
THEN EDIT FIELD:

Select ZZMY MULTIPLE NAME: MY TEST

>D ^%G

 Global ^DIZ(16029
        DIZ(16029
^DIZ(16029,0) = ZZMY MULTIPLE^16029^1^1
^DIZ(16029,1,0) = xyz           <== the .01 was changed
^DIZ(16029,"B","MY TEST",1) =   <== and now is not properly indexed
```

```
Global ^
```

This patch makes the 4-slash stuff add the entry into the multiple if it doesn't already exist, or select the entry if it does already exist.

78.1.3 Routine: DIEQ

3. Bug fixed: If you (1) have screen on a SET OF CODES field, (2) make a call ^DIE to edit that field for a given entry, (3) answer something wrong, and then (4) up-arrow out, DIC("S") will leak out of the call. For example:

```
16029,55      SCREENED SET           0;2 SET
                                  '1' FOR ONE;
                                  '2' FOR TWO;
              SCREEN:             S DIC("S")="I 1"
              EXPLANATION:        Test screen always evaluates to true.

>S DIE=16029,DR=55,DA=2 D ^DIE

SCREENED SET: ONE// THREE??
     Test screen always evaluates to true.
     Choose from:
       1        ONE
       2        TWO
SCREENED SET: ONE// ^

>ZW DIC("S")
DIC("S")=I 1
```

This patch makes sure DIC("S") is cleaned up after the call. The presence of DIC("S") could affect subsequent ^DIC calls.

79.0 DI*22*82 SEQ #79: Edit File Missing Attribute & Deletion Dialog

79.1 Description

79.1.1 Bug Fixes

1. While using the FileMan option UTILITY FUNCTIONS/EDIT FILE and Deleting a file, at the prompt 'IS IT OK TO DELETE THE '^DIZ(###)' GLOBAL?' no input is allowed and FileMan continues on and delete the Data Dictionary, Input Templates, etc.

This patch allows input at the above prompt and adds a safety measure by adding the prompt SURE YOU WANT TO DELETE THE ENTIRE FILE? After the above prompt IS IT OK TO DELETE THE '^DIZ(###)' GLOBAL?. This allows the user an opportunity to exit without deleting any portion of a file.

Routine: DIBT, DICAT, DICATT4, DINIT0FE, DINIT0FF, DIU0, DIU2, DIU20, DIU21

Form: DIEDIT

2. This patch also enhances Option UTILITY FUNCTIONS/EDIT FILE by adding FILE SCREEN to this option. MUMPS code can be entered in FILE SCREEN which would be a permanent DIC("S") for the file.

Routine: DIBT, DICAT, DICATT4, DINIT0FE, DINIT0FF, DIU0, DIU2, DIU20, DIU21

Form: DIEDIT

80.0 DI*22*41 SEQ #80: Re-Indexing & KIDS Fixes And New Entry Points

80.1 Description

80.1.1 Routines: DIK, DIK1, DIU1

1. **Bug fix**: When you use FileMan's RE-INDEX FILE option on the UTILITY FUNCTIONS menu, and you reindex an entry in a subfile, the entire index is not killed first. Therefore, bad nodes can survive a reindex. This is not the case when reindexing a top-level index. The IX^DIK and EN^DIK calls have the same problem. This patch makes FileMan issue a single Kill command to remove an entire index when all entries in that index are re-indexed.

2. **Bug fix**: If a file has only one cross-reference, and that cross-reference is a new-style cross-reference, an undefined DIUCNT(1) error would occur when you use FileMan's RE-INDEX FILE option on the UTILITY FUNCTIONS menu. (The error occurred before any reindexing actually took place, so database integrity is not affected.)

80.1.2 Routines: DIFROMS4, DINIT004, DINIT00X, DITR, DITR1

3. **Bug fix**: During a KIDS install, when data overwrote existing data, indexes on the old data were not killed, so you could end up with indexes on both old and new values of the same field.

4. **Bug fix**: During a KIDS install, when files with a .001 field were sent with data, the data was not being correctly filed. If a record with an IEN was sent, and an existing record with the same IEN was already on file but the .01 field or Identifiers didn't match, a new record was added at a different IEN. Instead, the incoming record should have been rejected.

The DINIT routine is being sent to set a new entry in the DIALOG file, an error message in case a record can't be updated during the KIDS build. However, the installer should not rerun ^DINIT. The new entry will be sent automatically as part of the KIDS build for this patch.

80.1.3 Routines: DIK, DIK1

5. New entry points:

80.1.3.1 IX2^DIK

Executes the KILL logic for only one entry at all file levels at or below the one specified in DIK.

Before calling this entry point, you should be familiar with the effects of executing the kill logic of the relevant cross-references (including bulletins, triggers, and MUMPS-type).

Input Variables

DIK If you are executing the kill logic for an entry at the top level of a file, set DIK to the global root of the file.

If you are executing the kill logic for a subentry, set DIK to the full global root leading to the subentry, including all intervening subscripts and the terminating comma, up to but not including the IEN of the subfile entry.

DA If you are executing the kill logic for an entry at the top level of a file, set DA to the internal entry number of that file entry.

If you are executing the kill logic for an entry in a subfile, set up DA as an array, where DA is the entry number in the subfile, DA(1) is the entry number at the next higher file level, etc. DA(n) is the entry number at the file's top level.

80.1.3.2 IXALL2^DIK

Executes the KILL logic for all entries in a file.

Before calling this entry point, you should be familiar with the effects of executing the kill logic of the relevant cross-references (including bulletins, triggers, and MUMPS-type).

> **NOTE**: IXALL^DIK, IXALL2^DIK, ENALL^DIK, ENALL2^DIK, and the Re-Index File option on the Utility Functions menu set the 3rd piece of the 0 node of the file's global root (the file header) to the last internal entry number used in the file. They set the 4th piece to the total number of entries in the file.

Input Variables

DIK If you are executing the kill logic for all entries at the top level of a file, set DIK to the global root of the file.

If you are executing the kill logic for all entries in a subfile only, set DIK to the full global root of the subfile.

DA If you are executing the kill logic for all entries at the top level of a file this variable need not be set.

If you are executing the kill logic for all entries in a subfile, set up DA as an array, where DA(1) is the entry number at the next higher file level, DA(2) is the entry number one level above that, etc. DA(n) is the entry number at the file's top level.

80.1.3.3 EN2^DIK

Executes the KILL logic for one or more cross-references on a specific field for one entry in a file.

Before calling this entry point, you should be familiar with the effects of executing the kill logic of the relevant cross-references (including bulletins, triggers, and MUMPS-type).

Input Variables

DIK If you are executing the kill logic for an entry at the top level of a file, set DIK to the global root of the file.

If you are executing the kill logic for a subentry, set DIK to the full global root leading to the subentry, including all intervening subscripts and the terminating comma, up to but not including, the IEN of the subfile entry.

DA If you are executing the kill logic for an entry at the top level of a file, set DA to the internal entry number of that file entry.

If you are executing the kill logic for an entry in a subfile, set up DA as an array, where DA is the entry number in the subfile, DA(1) is the entry number at the next higher file level, etc. DA(n) is the entry number at the file's top level.

DIK(1) Set DIK(1) to the field number (to get all cross-references defined on that field). For example:

```
S DIK(1)=.01
```

OR, set DIK(1) to the field number and the names or numbers of specific cross-reference on that field, all separated by up-arrows (^). For example,

```
S DIK(1)=".01^B"

S DIK(1)=".01^B^C"

S DIK(1)=".01^1^2"
```

80.1.3.4 ENALL2^DIK

Executes the KILL logic for one or more cross-references on a specific field for all entries in a file.

Before calling this entry point, you should be familiar with the effects of executing the kill logic of the relevant cross-references (including bulletins, triggers, and MUMPS-type).

NOTE: IXALL^DIK, IXALL2^DIK, ENALL^DIK, ENALL2^DIK, and the Re-Index File option on the Utility Functions menu set the 3rd piece of the 0 node of the file's global root (the file header) to the last internal entry number used in the file. They set the 4th piece to the total number of entries in the file.

Input Variables

DIK If you are executing the kill logic for all entries at the top level of a file, set DIK to the global root of the file.

If you are executing the kill logic for all entries in a subfile only, set DIK to the full global root of the subfile.

DA If you are executing the kill logic for all entries at the top level of a file, this variable need not be set.

If you are executing the kill logic for all entries in a subfile, set up DA as an array, where DA(1) is the entry number at the next higher file level, DA(2) is the entry number one level above that, etc. DA(n) is the entry number at the file's top level.

DIK(1) Set DIK(1) to the field number (to get all cross-references defined on that field). For example:

S DIK(1)=.01

OR, set DIK(1) to the field number and the names or numbers of specific cross-reference on that field, all separated by up-arrows (^). For example,

S DIK(1)=".01^B"

S DIK(1)=".01^B^C"

S DIK(1)=".01^1^2"

80.1.4 Routines: DIPR41, DINIT013

6. Pre-Install routine DIPR41 adds two new entries to the LANGUAGE file, 12 GREEK and 18 HEBREW.

The DINIT routine is being changed to set the new entries in the LANGUAGE file. However the installer should not rerun ^DINIT. The new entries are installed via the Pre-Install routine.

81.0 DI*22*68 SEQ #81: Key And Index Fixes

81.1 Description

81.1.1 Routine: DIKCFORM, DIKCUTL2

1. **Bug fixed**: When you add or delete field-type cross-reference values from a new-style cross-reference, the Execution of the cross-reference should be updated automatically to 'Field' if there is only one field-type cross-reference value, and to 'Record' if there are more than one field-type cross-reference values. Previously, the Execution field wasn't updated at all for MUMPS-type cross-references. For Regular cross-references, the Execution field was updated based on the total number of values, field or computed, in the cross-reference, rather than just the number of field-type values.

2. **Bug fixed**: If you edit a new-style cross-reference, and enter an at-sign (@) at the "Index Name" prompt, a null subscript error would occur. This patch makes FileMan display the message "Index Name is a required field" when you attempt to delete the Index Name.

81.1.2 Routine: DIKKUTL

3. **Bug fixed**: If you edit a key (FileMan menu -> Utility Functions -> Key Definition), and on the ScreenMan form, press <RET> at the "Index Details..." field, and edit the properties of the uniqueness index (either the name of the index or the length of its subscripts), FileMan wasn't automatically rebuilding the uniqueness index.

81.1.3 Routine: DIKKUTL1

4. **Typo fixed**: If you edit an existing key by changing the fields in that key, for example, after you exit the ScreenMan form, FileMan issues the following prompt:

```
The Key fields and the fields in the Uniqueness Index don't match.
  Select one of the following:
    1  Re-Edit the Key
    2  Make Key match Uniqueness Index (also selected on up-arrow)
    3  Make Uniqueness Index match Key
Enter response:
```

If you selected option 3 -- "Make Uniqueness Index match Key" – FileMan printed the message: "Modifing Uniqueness Index ...." This should be "Modifying Uniqueness Index ...."

81.1.4 Routine: DIKC

5. **Bug fixed**: If a computed-type cross-reference value in a new-style cross-reference killed X2, an undefined error would occur when any of the field-type cross-reference values in that cross-reference were edited. For example, suppose the cross-reference looks like:

```
     C (#268)    RECORD    REGULAR    IR    LOOKUP & SORTING
        Short Descr:  This xref contains a field and a computed subscript
                      that calls ^%DTC.
          Set Logic:  S ^DIZ(16033,"C",X(1),X(2),DA)=""
         Kill Logic:  K ^DIZ(16033,"C",X(1),X(2),DA)
         Whole Kill:  K ^DIZ(16033,"C")
               X(1):  FREE  (16033,11)  (Subscr 1)  (forwards)
               X(2):  Computed Code: S X1=3010701,X2=3010101 D ^%DTC
                        (Subscr 2)
```

The computed subscript in this cross-reference calls ^%DTC, which takes as input variables X1 and X2, and returns X. ^%DTC kills X1 and X2. If you use FileMan's "ENTER OR EDIT FILE ENTRIES" option to edit field #11, an undefined error would occur.

6. **Bug fixed**: If one process has a record in a file locked (for example, because a record in that file is being edited via FileMan's Enter/Edit option), and a user in another process uses FileMan's RE-INDEX FILE option on the UTILITY FUNCTIONS menu to reindex a new-style cross-reference defined on that same file, the reindexing operation appears to work, but because a record is locked, in actuality, the cross-reference is not rebuilt.

This patch makes it so that if a new-style cross-reference is reindexed, FileMan attempts a lock, but continues even if the lock cannot be obtained.

81.1.5 Routine: DIKCUTL1

7. **Bug fixed**: When you create a new-style cross-reference, FileMan presents you with a ScreenMan form where you can define the properties of the cross-reference. If you quit the form (that is, exit the form without saving changes), an incomplete cross-reference would be defined. This patch makes FileMan delete the cross-reference if you quit the form without defining any properties.

81.1.6 Routines: DIKCUTL3, DIKKUTL

8. **Bug fixed**: After you edit, delete, or create a new-style cross-reference, FileMan issues the prompt:

```
Press RETURN to continue:
```

This prompt is unnecessary and is removed by this patch.

The prompt also appears if you edit a key, select an existing new-style index as the uniqueness index such that the fields in the "KEY FIELDS" section of the ScreenMan form do not match the fields in the uniqueness index, exit the form, and then select the option "Make Key match Uniqueness Index."

81.1.7 Routines: DIKCUTL, DIKCUTL3

9. The changes to these two routines clarify two prompts issued when a new-style index is selected for deletion. Before, FileMan asked two questions (if you answered 'YES' to the first):

```
     Are you sure you want to delete the Index? No// YES
     Do you want to delete the old index now? YES//

   This patch changes the questions to:

     Are you sure you want to delete the Index definition? NO//
     Do you want to delete the data in the old index now? YES//
```

81.1.8 Routine: DIKD

10. **Bug fix**: When the DELIX^DDMOD API is used to delete a SOUNDEX cross-reference, the ^DD(file#,0,"LOOK") and ^DD(file#,0,"QUES") nodes weren't getting cleaned up. This patch ensures that those nodes are killed when the SOUNDEX index is deleted.

82.0 DI*22*49 SEQ #82: Undefined at 3+1~DIET

82.1 Description

82.1.1 Bug Fix

If the DG array is partially defined for some reason when adding a new record (FILE^DICN) and Auditing is turned on for the .01 field then an Undefined will occur at 3+1^DIET. This patch corrects this possibility by Newing the DG array.

Routine: DIET

83.0 DI*22*90 SEQ #83: Validating Variable Pointers

83.1 Description

83.1.1 Routine: DIEV1

CHK^DIE and VAL^DIE rejected the external value of a variable pointer field if it found an exact match in the "B" index of a pointed-to file, but also found an exact match to another entry in another index in that file. For example, if "EXERCISE" is in both the "B" index and the "C" index of the pointed-to file, and the IENs of the two entries are different, the value "EXERCISE" is rejected.

This patch makes CHK^DIE and VAL^DIE look only in the "B" index of the pointed to file(s) for a match to a variable pointer.

This change also affects the Filer (FILE^DIE) and the Updater (UPDATE^DIE) when the "E" flag is used (i.e., external values are passed). The Updater and Filer will now look for matches to the variable pointer only in the "B" index of the pointed to file(s).

This patch, in effect, backs out the last change described for routine DIEV1 in patch DI*22*72.

84.0 DI*22*94 SEQ #84: Recompiling Forms During a KIDS Install

84.1 Description

84.1.1 Routine: DDSZ, DIFROMSI

Bug fixed: During a KIDS installation at a site, KIDS automatically compiles any ScreenMan forms that are transported. However, if a transported form contains blocks that are used on other ScreenMan forms, those forms should also be recompiled. This patch makes KIDS recompile all forms that contain any of the blocks used on a form being sent in a KIDS install.

The post-install routine DIPS94 recompiles the ScreenMan forms that were affected by patch XU*8*214. That patch sent the XUEXISTING USER form, which should also have caused an automatic recompilation of the forms XUREACT USER, XU-PERSON CLASS, and XUNEW USER.

85.0 DI*22*92 SEQ #85: Transporting Fields That Are Part Of New-Style XREFS

85.1 Description

85.1.1 Routines: DIFROMSX

1. **Change**: Before this patch, when you used KIDS to transport a Partial DD and attempted to transport a field that is part of a new-style cross-reference, KIDS would abort the transport and print out an error message if you failed to transport all of the fields that take part in that cross-reference. With this patch, you can transport only one or some of the fields that are part of a compound cross-reference.

85.1.2 Routine: DIFROMSY

2. **Change**: Before this patch, when you used KIDS to transport a Partial DD and attempted to transport a field that is part of a compound key (a key that is composed of more than one field), KIDS would abort the transport and print out an error message if you failed to transport all of the fields that take part in that key. With this patch, you can transport only one or some of the fields that are part of a compound key.

86.0 DI*22*83 SEQ #86: Corrupted Field Zeroth Node

86.1 Description

86.1.1 Repair

This patch fixes one piece of the zeros node of FileMan field definition from being corrupted resulting in a bad variable name error. This error will only happen, If And Only If, when a User tries to use certain types of FM field modified in the following manner:

If a user defined the field to be either a Pointer or Set of Codes and then later at some point time decided to use the Screen-Mode version of Modify File Attributes and arrowed down to make the field Mandatory, all of the Pointer or Set of Codes information would be deleted.

For example, the '%DSM-E-NAME, bad variable name' hard error would occur in routine DIED during usage.

* Please note that the patch fixes the corruption in future but users might run into some corrupted definitions created under the conditions mentioned above resulting in hard error mentioned in the above example.

A FM Patch in near future will search for these problem nodes and report the corrupted nodes. In the meantime, if Fields with this problem are found, these can be corrected by using Modify File Attributes, stepping through all fields (especially the DATA TYPE field) and saving the modifications.

Routine: DICATTD8, DICATTDE

87.0 DI*22*91 SEQ #87: Printing of Word Processing Fields

87.1 Description

87.1.1 Repair

If a user wanted to print a multi-lined Word Processing field using the "X" suffix (Suppress header and inter-column spaces), followed by another field or literal, then the field or literal would print after the first line of the Word Processing field is printed. For example:

```
OUTPUT FROM WHAT FILE: ZZSO//
STANDARD CAPTIONED OUTPUT? Yes// N  (No)
FIRST PRINT FIELD: .01  NAME
THEN PRINT FIELD: 11;X;R80  WP FIELD
THEN PRINT FIELD: DATE
DEVICE:   IRMS BLDG 74 ROOM 227E    Right Margin: 80//
ZZSO LIST                                  FEB  5,2002  12:50    PAGE 1
NAME                            DATE
------------------------------------------------------------------------------

AEFMTUFJO,UIPNBT F
ID,NPO AND CONSENT VERIFIED W/ PT.  OPERATION SITE VERIFIED AND MARKED BY
MAY 30 ,2001  <<this is the problem
SURGEON ALLERGY TO IV IODINE FOLEY CATH.FR.18-5CC INSERTED BY DR. X
<and so on>

Using the same example after the patch, the report would look like:

ZZSO LIST                                  FEB  5,2002  12:57    PAGE 1
NAME
  DATE
------------------------------------------------------------------------------

AEFMTUFJO,UIPNBT F
ID,NPO AND CONSENT VERIFIED W/ PT.  OPERATION SITE VERIFIED AND MARKED BY
SURGEON ALLERGY TO IV IODINE FOLEY CATH.FR.18-5CC INSERTED BY DR. X
<and so on>

  MAY 30,2001
```

Routine: DIL0

88.0 DI*22*89 SEQ #88: Undefined S~DICATT2

88.1 Description

This patch addresses the following issues:

1. Routine DIPR89 will run at the beginning of the install. It checks for possible corrupted DD field nodes and reports during Installation (Patch DI*22*83 stopped further corruption). If no corrupted DD field nodes are found, you will notice the message, 'No problems found' and no further action is necessary.

   If any such nodes are found, the following information will be written into the Install file: (For example)

```
>>File/Subfile: 662055
  Field: #7(TEST SET OF CODES)  Type: Set
  Node=TEST 83 SET^RS^^5;2^Q
```

   Please log a NOIS to get further guidance on how to fix these nodes. Routine DIPR89 will be left on the target system as a courtesy, just incase the site would like to use it again by doing - D ^DIPR89.

2. Fixes an undefined error which occurs in DICATT2 when a user, using Modify File Attributes and adds a new field of any Type using Screen-Mode version or 'roll and scroll' but does not follow through and presses an '^' at the DEVICE prompt. Here are the steps to reproduce the error:

   a. Add a new field of any Type (Screen-Mode version or 'roll and scroll'.

   b. Then using 'roll and scroll' change the Type.

   c. When asked:

      SINCE YOU HAVE CHANGED THE FIELD DEFINITION,

      EXISTING 'TEST TWO' DATA WILL NOW BE CHECKED FOR

      INCONSISTENCIES OK? Yes// (Yes)

   d. At the device prompt use up-arrow: DEVICE: HOME// ^

   e. Then using the 'roll and scroll' re-step through the field definition, the error will occur after the following prompt is displayed:

      (OPTIONAL) PATTERN MATCH (IN 'X'):

89.0 Contact Information

If you have any questions or comments regarding this distribution, please contact the ITSC Help Desk by:

Phone: (505) 248-4371 or
(888) 830-7280

Fax: (505) 248-4199

Web: http://www.rpms.ihs.gov/TechSupp.asp

Email: RPMSHelp@mail.ihs.gov